Network & Communication

NETWORK & COMMUNICATION	
hostname	prints the computer's hostname, **-s** short hostname (without the domain name), **-I** all IP addresses of the computer, **-d** DNS domain name, **-y** NIS domain name
hostname <hostname>	sets the computer's hostname (permanent settings are configured in */etc/sysconfig/network*)
hostnamectl (implemented from RHEL 7)	[**status**] prints the computer's hostname, hardware type, machine ID, boot ID, operating system name, kernel version and processor architecture, **hostname** only prints the computer's hostname
hostnamectl set-hostname | **hostname** <hostname> (implemented from RHEL 7)	sets a permanent computer's hostname (edits */etc/hostname*), **--transient** sets a temporary computer's hostname
domainname	prints the computer's NIS domain name, **-I** all IP addresses of the computer, **-d** DNS domain name
domainname <nisdomain>	sets the computer's NIS domain name (permanent settings are configured in */etc/sysconfig/network*)
dnsdomainname | **domainname -d**	prints the computer's DNS domain name
hostid	prints the numeric identifier of the computer
cat /etc/services	prints known network services including their protocol and port number
cat /etc/protocols	prints known network protocols
cat /etc/resolv.conf	prints a list of available DNS servers to resolve domain names to IP addresses
cat /etc/hosts	prints a list of IP addresses and their associated hostnames, including aliases, used to resolve hostnames to the corresponding IP addresses locally without the need for querying a DNS server
host [<IP_address | hostname>] [<DNS_server>]	prints the name or IP address of the remote host and possibly also the IP address of the DNS server used, **-t** <type> specifies the query type ("A", "AAAA" and "MX" by default), **-v** detailed output; a specified DNS server can be used for DNS lookups instead of the servers configured in */etc/resolv.conf* `$ host prompt.cz` (prints the IPv4 and IPv6 address of the remote host and the name of the domain's mail server)
nslookup [<IP_address | hostname>] [<DNS_server>]	prints the name or IP address of the remote host and possibly also the IP address of the DNS server used, **-query=**<type> specifies the query type ("A" and "AAAA" by default); a specified DNS server can be used for DNS lookups instead of the servers configured in */etc/resolv.conf*; with no argument it starts in interactive mode
dig [<hostname>] [@<DNS_server>]	prints the IP address of the remote host and the IP address of the DNS server used, **-x** <IP_address> prints the name of the remote host, **-t** <type> specifies the query type ("A" by default); a specified DNS server can be used for DNS lookups instead of the servers configured in */etc/resolv.conf*
whois [<domain_name>]	prints information about the registration of an internet domain (the domain name, organization that registered the domain, registration date, expiration date, owner and contacts) `$ whois prompt.cz` (prints information about the registration of the internet domain)
arp [<IP_address | hostname>]	displays entries in the ARP table (mappings of IP addresses to their corresponding MAC addresses for network devices that the system has recently communicated with within the same subnet), **-n** does not resolve hostnames via DNS, **-i** <device> specifies the network interface, **-s** <IP_address> <MAC_address> adds an entry to the ARP table, **-d** <IP_address> deletes an entry from the ARP table
ping [<IP_address | hostname>]	detects the reachability of a host on the network by sending "ICMP echo request" packets and receiving "ICMP echo reply" packets, printing the number of packets sent, received and lost along with the response time for each packet, **-b** uses a broadcast address, **-c** <n> specifies the number of packets to be sent to the target host, **-i** <n> specifies the interval between sending each packet in seconds (1 s by default), **-I** <device> specifies the network interface for sending the packets, **-n** does not resolve hostnames via DNS, **-W** <n> specifies the response timeout in seconds `$ ping -c 5 prompt.cz` (detects the reachability of a host on the network by sending five "ICMP echo request" packets)

NETWORK & COMMUNICATION	
route [<command>] [<destination>] [<specification ...>]	prints the kernel's IP routing table, **-n** does not resolve hostnames via DNS, **add** adds a static route, **del** removes a static route, **-net** specifies a network, **-host** specifies a host, **netmask** specifies a network mask, **gw** specifies a gateway, **dev** specifies a network interface; (permanent settings are configured in */etc/sysconfig/network-scripts/route-**) `# route add default gw 192.168.1.1` (adds a default route via the gateway "192.168.1.1") `# route add -host 192.168.100.10 gw 192.168.100.1` (adds a route to the host "192.168.100.10" via the gateway "192.168.100.1") `# route add -net 192.168.100.0/24 gw 192.168.100.1 dev eth1` (adds a route to the network "192.168.100.0/24" via the gateway "192.168.100.1" and the network interface "eth1") `# route del -net 192.168.100.0/24 gw 192.168.100.1 dev eth1` (removes a route to the network "192.168.100.0/24" via the gateway "192.168.100.1" and the network interface "eth1")
traceroute [<IP_address | hostname>]	prints the network path to a remote host, **-m** <n> specifies the max. number of hops – max. time-to-live value (30 by default), **-n** does not resolve hostnames via DNS, **-i** <device> specifies the network interface for sending the packets, **-w** <n> specifies the response timeout in seconds (5 s by default) `$ traceroute prompt.cz`
tracepath [<IP_address | hostname>]	prints the network path to a remote host, **-m** <n> specifies the max. number of hops – max. time-to-live value (30 by default), **-n** does not resolve hostnames via DNS, **-b** prints both hostnames and IP addresses
mtr [<IP_address | hostname>]	displays the network path to a remote host in an interactive and dynamic way, including packet loss percentage, the number of packets sent and response time for each hop, **-m** <n> specifies the max. number of hops – max. time-to-live value (30 by default), **-n** does not resolve hostnames via DNS, **-I** <device> specifies the network interface for sending the packets; interactive options: **p** pauses the current display, **q** quits the program
ip [<object>] [<command>] [<destination>] [<specification ...>]	prints or configures network parameters of the specified object, **addr** {**add** | **del** | **show**} adds, removes or displays IP addresses of a network interface, **link** {**add** | **del** | **set** | **show**} adds, removes, sets or displays properties of a network interface, **neigh** {**add** | **del** | **change** | **show**} adds, removes, changes or displays entries in the ARP table, **route** {**add** | **del** | **change** | **show**} adds, removes, changes or displays entries in the kernel's IP routing table, **-s** displays traffic statistics for a network interface; (permanent settings are configured in */etc/sysconfig/network-scripts/*) `$ ip link show` (prints the properties of all network interfaces – their status, MAC address and other network parameters) `$ ip -s link show enp3s0` (prints the properties of a specified network interface, including information about the number of packets and bytes sent and received) `# ip link set down enp1s6` (deactivates a network interface) `$ ip addr show` (prints the properties of all network interfaces – their status, MAC address, IP address, network mask and other network parameters) `# ip addr add 192.168.0.100/24 dev eth0` (adds an IP address to the network interface) `# ip addr del 192.168.0.100/24 dev eth0` (removes an IP address from the network interface) `$ ip neigh show` (prints the ARP table) `$ ip route show` (prints the kernel's IP routing table) `# ip route add default via 192.168.1.1` (adds a default route via the gateway "192.168.1.1") `# ip route add 192.168.100.10 via 192.168.100.1` (adds a route to the host "192.168.100.10" via the gateway "192.168.100.1") `# ip route add 192.168.100.0/24 via 192.168.100.1 dev eth1` (adds a route to the network "192.168.100.0/24" via the gateway "192.168.100.1" and the network interface "eth1") `# ip route del 192.168.100.0/24 via 192.168.100.1 dev eth1` (removes a route to the network "192.168.100.0/24" via the gateway "192.168.100.1" and the network interface "eth1")
ifconfig [<device>]	prints the properties of all active or specified network interfaces – their status, MAC address, IP address, network mask and other network parameters, **-a** prints also inactive network interfaces
ifconfig <device> <specification ...>	configures a specified network interface, **up** activates a device, **down** deactivates a device, **hw ether** <MAC_address> specifies a MAC address, **netmask** <netmask> specifies a network mask, **mtu** <n> specifies the maximum transfer unit; (permanent settings are configured in */etc/sysconfig/network-scripts/ifcfg-**) `# ifconfig eth0 down; ifconfig eth0 up` (deactivates and activates a network interface) `# ifconfig eth0 192.168.0.10 netmask 255.255.255.0` (sets a static IP address and network mask for the network interface) `# ifconfig eth0 hw ether 00:11:09:D6:DC:3C` (sets a specified MAC address for the network interface)

NETWORK & COMMUNICATION	
ifup <device>	activates a network interface `# ifup eth0`
ifdown <device>	deactivates a network interface `# ifdown eth1`
iwconfig [<device>] [<specification ...>]	configures or prints the properties of a wireless network interface, **essid** <network_name> specifies the network name, **ap** <AP_address> specifies the address of the access point to which the interface should connect, **mode** <mode> specifies the operating mode of the device ("Managed" = client, "Master" = access point), **key** <key> specifies the encryption key `# iwconfig eth1 essid AP_profik ap 00:60:1D:01:23:45 key 0123-4567-89 mode Managed` (configures the wireless interface "eth1" to connect to the network "AP_profik", with the access point address "00:60:1D:01:23:45", using the encryption key "0123-4567-89" for the secure connection, and in operating mode "Managed")
iwlist [<device>] [<parameter>]	prints detailed information about wireless network interfaces and networks, **scan** prints available wireless networks, including IP addresses of access points, frequency, mode, encryption and quality
nmcli [<object> <command> [<argument ...>]]	prints information about network devices and network configuration, **con** {**add** | **del** | **show** | **mod** | **reload** | **up** | **down**} creates, deletes, displays, modifies, reloads, activates or deactivates a network connection profile, **dev** {**con** | **dis** | **status** | **show** | **wifi**} connects, disconnects, displays network device status or lists available Wi-Fi access points, **radio wifi** [on | off] displays or sets Wi-Fi status; the network connection profile is a collection of settings that can be configured for a specified device, each profile having a name or ID that identifies it `$ nmcli dev status` (displays the status of all network devices) `$ nmcli dev show enp3s0` (displays detailed information about the specified network device) `$ nmcli con show` (displays all network connection profiles) `$ nmcli con show --active` (displays only active network connection profiles) `$ nmcli con show enp3s0` (displays detailed information about the specified network connection profile) `# nmcli con add con-name static ifname enp3s0 type ethernet ipv4.method manual ipv4.address 192.168.15.105/24 ipv4.gateway 192.168.15.1 ipv4.dns 192.168.15.1` (creates a new network connection profile "static" with a specified IP address, network prefix, default gateway and DNS) `# nmcli con mod static +ipv4.address 192.168.15.106/24` (modifies the network connection profile by adding another IP address) `# nmcli con mod static +ipv4.routes 192.168.15.0/24 10.10.10.1` (modifies the network connection profile by adding another static route) `# nmcli con up static` (activates the network connection profile) `# nmcli con mod static connection.id primary` (renames the network connection profile to "primary") `# nmcli con mod enp3s0 autoconnect no` (disables the original network connection profile from autostarting at boot) `# nmcli con reload` (reloads the configuration file changes) `# nmcli con down enp3s0` (deactivates the network connection profile) `# nmcli con del enp3s0` (deletes the network connection profile) `# nmcli dev dis enp3s0` (disconnects the specified network device) `$ nmcli radio wifi on` (enables Wi-Fi connection) `$ nmcli dev wifi list` (lists available Wi-Fi networks) `$ nmcli dev wifi connect WiFi01` (connects to the Wi-Fi network specified by the SSID)
whatmask [<netmask | IP_address/netmask>]	prints the number of usable IP addresses in a specified network; if an IP address is provided along with the network mask, the network address, broadcast address, and the first and last usable IP addresses are printed as well `$ whatmask /24` | `$ whatmask 255.255.255.0` (prints the number of usable IP addresses in the network) `$ whatmask 192.168.15.100/24` | `$ whatmask 192.168.15.100/255.255.255.0` (prints the number of usable IP addresses in the network, including the network address, broadcast address, and the first and last usable IP addresses)
ethtool [<device>]	prints the ethernet card settings, **-S** displays network traffic statistics, **-s** <device> <parameter ...> modifies the ethernet card settings – **duplex** {full | half} sets full or half duplex mode, **speed** <n> sets the speed in Mb/s `# ethtool -s eth0 duplex full speed 100` (configures the ethernet card to operate in full duplex mode at a speed of 100 Mb/s)

NETWORK & COMMUNICATION	
ifstat [<device>]	displays network traffic statistics – the size of received and transmitted data on all or specified network interfaces (only the difference from the previous display is printed)
iftop	displays network traffic between remote hosts in an interactive and dynamic way – the source and destination addresses, rate at which data has been sent and received over the preceding 2, 10 and 40 second intervals and the total summary, **-i** <device> specifies a network interface (the first one by default); interactive options: **n** does not resolve hostnames via DNS, **p** displays ports, **N** does not resolve port numbers to service names, **P** pauses the current display, **q** quits the program
tcpdump [<expression>]	displays traffic on a network, **-i** <device> on a specified interface (the first one by default), **port** <port> on a specified port, **tcp** | **udp** | **icmp** for a specified protocol, **host** <host> between a specified host, **ether host** <MAC_address> between a specified MAC address, **-n** does not resolve hostnames via DNS, **-r** <file> reads packets from a file, **-w** <file> writes packets to a file, **-X** displays the data of each packet in hexadecimal and ASCII format, **-v** detailed output `# tcpdump -i eth0 -nv port 22` (displays network traffic on the device "eth0" and port 22) `# tcpdump -nv ether host 00:02:3F:09:FA:F1` (displays network traffic on the device with the specified MAC address) `# tcpdump -X host prompt.cz` (displays network traffic between the specified host in hexadecimal and ASCII format)
netstat | **ss**	displays a list of open sockets, including the protocols used, local and remote addresses, and connection states, **-a** all current connections, **-l** listening ports, **-t** TCP ports (along with the "-a" or "-l" option), **-u** UDP ports (along with the "-a" or "-l" option), **-e** users and i-nodes, **-p** the PID and name of the program associated with each connection, **-i** the table of network interfaces (only applies to the "netstat" command), **-r** the kernel's IP routing table (only applies to the "netstat" command), **-s** summary statistics for each protocol, **-n** displays port numbers instead of service names (and computers' IP addresses and user UIDs instead of their names when using the "netstat" command) `# netstat -tupan` (prints active TCP and UDP network connections, including listening ports and programs associated with them)
ncat [<host>] [<port>]	reads and writes data across networks, **-e** <command> executes the specified command, **-l** <port> starts listening on the specified port, **-n** does not resolve hostnames via DNS, **-u** uses a UDP connection instead of default TCP, **-z** scans for open ports without sending any data, **-v** detailed output `$ ncat -zv prompt.cz 80` (scans for the availability of port 80 on the remote host) `$ ncat -l 1234 -e /bin/bash` (starts listening on port 1234 and opens a shell) `$ ncat 192.168.124.80 1234` (connects to the host on port 1234 with the possibility of remote command execution) `$ ncat -l 1234 > data.txt` (starts listening on port 1234 and writes the received data to a file) `$ cat data.txt | ncat 192.168.124.80 1234` (transfers data to the remote host on port 1234)
nmap [<scan>] [<host ...>]	scans the availability of ports on a host in order to identify running services, **-sS** performs a TCP SYN scan (the most used scan, does not open a full TCP connection, sends a SYN packet and receives SYN/ACK – port is open, or RST – port is closed), **-sT** performs a TCP connect scan (opens a full TCP connection), **-sU** performs a UDP scan, **-sn** only detects a host's availability in the network, in the local network its MAC address is also displayed, **-Pn** does not detect a host's availability in the network, **-O** detects the operating system type, **-sV** detects the version of the service, **-D** <IP_address>[,<IP_address>...] specifies a list of decoy hosts to make it appear that the target host is being scanned by multiple systems simultaneously in addition to the real source IP address, **-iL** <file> reads the target hosts from a file, **-n** does not resolve hostnames via DNS, **-p** <port> specifies a range of ports (1000 most common ports by default), "**-**" specifies all 65535 ports, **-v** detailed output `$ nmap prompt.cz` (scans most common TCP ports on the target host) `# nmap -p - -sS -sU localhost` (scans all TCP and UDP ports on the local host) `# nmap -sn 192.168.15.0/24` (prints all available hosts on the network) `# nmap -sS -sV 147.229.28.4 > scan.txt` (starts a TCP SYN scan with service version detection and saves the result to the specified file) `# nmap -sS -Pn -p 1-1023 192.168.0.247` (starts a TCP SYN scan without a ping request and specifies the port range) `# nmap -sS -sU -iL hosts` (starts a TCP SYN and UDP scan on the hosts specified in the file) `# nmap -sS -O -D 192.168.0.1,192.168.0.2 192.168.0.3` (starts a TCP SYN scan with OS detection on "192.168.0.3" spoofed from the specified IP addresses)
service iptables <command>	**start** starts the firewall, **stop** stops the firewall, **restart** restarts the firewall, **status** prints the firewall settings, **save** saves the newly created firewall rules to */etc/sysconfig/iptables* to be preserved after the system reboot

NETWORK & COMMUNICATION	
iptables-save	exports configured (even unsaved) firewall rules from memory to STDOUT `# iptables-save > iprules` (saves the new firewall rules to the specified file)
iptables-restore	imports firewall rules from STDIN to memory `# iptables-restore < iprules` (reads the firewall rules from the specified file)

NETWORK & COMMUNICATION	
iptables [<chain>] [<specification>] [<target>]	configures firewall rules that control incoming and outgoing network traffic, **-t** <table> specifies the table where the rule should be applied; the "filter" table (default) is used to filter packets and contains the built-in chains "INPUT" for incoming packets destined for the local system, "OUTPUT" for outgoing packets generated by the local system and "FORWARD" for packets that are being routed through the system; the "nat" table is used to translate private network IP addresses and port forwarding with the built-in chains "PREROUTING" for altering incoming packets as soon as they come in, "INPUT" for altering incoming packets destined for the local system, "OUTPUT" for altering locally-generated packets before routing and "POSTROUTING" for altering outgoing packets as they leave the system; the "mangle" table is used for specialized packet alterations and contains all the above built-in chains; the "raw" table is used for configuring exemptions from connection tracking and provides the built-in chains "PREROUTING" and "OUTPUT"; and the "security" table is used for Mandatory Access Control (MAC) networking rules with the built-in chains "INPUT", "OUTPUT" and "FORWARD", **-I** <chain> [<rule_number>] inserts a rule to the beginning of the chain or to the specified position, **-A** <chain> appends a rule to the end of the specified chain, **-D** <chain> <rule_number> removes a rule from the specified chain, **-L** [<chain>] lists all rules in the particular chain, if no chain is specified, all chains are listed; with the **-n** option IP addresses and ports are printed in numeric format, with the **-v** option the number of packets and bytes for each rule including the protocol and interface are printed, with the **--line-numbers** option line numbers are added to the beginning of each rule for the particular chain (useful for further use with the "-I" or "-D" option), **-F** [<chain>] removes the rules from the particular chain, if no chain is specified, all rules are removed, **-P** <chain> <target> sets the default policy for the chain (all is allowed by default), **-N** <chain> creates a user-defined chain by the specified name, usually used for more detailed specifications of the rules (the default policy cannot be applied for these chains), **-X** <chain> removes a user-defined chain; the rule specification includes: **-i** <interface> input interface, **-o** <interface> output interface, **-s** <address> source address, **-d** <address> destination address, **-p** <protocol> type of protocol, **-m** <module> rule extension (**state** --state <connection_type> specifies the connection type – NEW new connection, ESTABLISHED existing connection, RELATED new connection related to an already existing communication, INVALID invalid connection when the packets cannot be identified; **time** specifies the time of connection --timestart <hh:mm>, --timestop <hh:mm>, --monthdays <day_in_month>, --weekdays <day_in_week>; **iprange** --src-range | --dst-range <IP_address>-<IP_address> specifies the range of source or destination IP addresses; **limit** --limit <n>/{s | m | h | d} specifies the time value, --limit-burst <n> specifies the number of packets), **--sport** <port> source port, **--dport** <port> destination port; and finally **-j** <target> specifies how to handle the packets – for the "filter" table ACCEPT = accept, DROP = drop, LOG = log the packets, REJECT = send back an error packet in response to the matched packet, for the "nat" table SNAT --to <IP_address> = change the source IP address, DNAT --to <IP_address> = change the destination IP address, REDIRECT --to-ports <port> = redirect the port; correct firewall settings strictly depend on the specific order of the rules listed in */etc/sysconfig/iptables* `# iptables -nvL --line-numbers` (prints the firewall rules in detailed output) `# iptables -P INPUT DROP` (drops all incoming packets) `# iptables -I INPUT -s 147.229.28.4 -j DROP` (drops packets coming from the specified IP address) `# iptables -A INPUT -p tcp --dport 22 -j DROP` (drops packets coming to the specified port) `# iptables -A INPUT -p tcp --dport 443 -j REJECT` (sends information about the service unavailability) `# iptables -I OUTPUT -d '!' 147.229.28.4 -j DROP` (allows only packets outgoing to the specified IP address) `# iptables -A OUTPUT -o eth0 -d 192.168.0.0/24 -j ACCEPT` (allows only packets outgoing from the specified interface to the local network) `# iptables -A OUTPUT -d upc.cz -p tcp --dport 80 -j DROP` (disallows to display the specified URL) `# iptables -A FORWARD -i eth0 -o eth1 -p tcp --dport '!' 80 -j DROP` (allows packet to be forwarded only to port 80) `# iptables -A INPUT -p tcp –dport 50:55 -m iprange --src-range 192.168.0.1-192.168.0.10 -j ACCEPT` (allows port range 50-55 for the source IP addresses) `# iptables -A INPUT -p icmp --icmp-type echo-request -m limit --limit 1/s --limit-burst 2 -j ACCEPT` (limits the number of "ping" requests to 2 per 1s) `# iptables -t nat -A PREROUTING -p tcp --dport 80 -j REDIRECT --to-ports 3250` (redirects destination port 80 to 3250) `# iptables -t nat -A PREROUTING -p tcp --dport 80 -i eth0 -j DNAT --to 192.168.0.2:8080` (alters the destination IP address and port of the service) `# iptables -A INPUT -j LOG` (logs all packets that do not match any of the configured rules to */var/log/messages*) `# iptables -D INPUT 5` (removes the rule from the "INPUT" chain found in the fifth position)

NETWORK & COMMUNICATION	
firewall-cmd [<specification>] (implemented from RHEL 7)	configures firewall rules that control incoming and outgoing network traffic, **--get-default-zone** prints default zone for network connections and interfaces, **--set-default-zone=**<zone> sets default zone for network connections and interfaces, **--get-active-zones** prints currently active zones altogether with network interfaces and sources used in these zones, **--get-zones** prints all available zones, **--list-all-zones** prints detailed information about all zones, **--zone=**<zone> specifies a zone (if not specified, the default zone is used), **--list-all** prints detailed information about the zone, **--get-services** prints all available services, **--list-services** prints services added to the zone, **--list-ports** prints ports added to the zone, **--add-source=**<IP_address>[/<netmask>] routes all traffic coming from the IP address or network to the zone, **--remove-source=**<IP_address>[/<netmask>] removes the rule routing all traffic coming from the IP address or network from the zone, **--add-interface=**<interface> routes all traffic coming from the network interface to the zone, **--change-interface=**<interface> changes the network interface for the zone, **--add-service=**<service> adds a service to the zone, **--remove-service=**<service> removes a service from the zone, **--add-port=**<port>/<protocol> adds a port to the zone, **--remove-port=**<port>/<protocol> removes a port from the zone, **--add-rich-rule=**<rule> adds a custom firewall rule to the zone that is not covered by the basic firewalld syntax, **--remove-rich-rule=**<rule> removes a custom firewall rule from the zone, **--query-rich-rule=**<rule> verifies if a custom firewall rule has been added to the zone, **--list-rich-rules** prints all custom firewall rules for the zone, **--permanent** performs a permanent configuration (writes changes to */etc/firewalld/*), **--reload** applies the permanent configuration, **--runtime-to-permanent** converts the current runtime configuration to permanent `# firewall-cmd --add-service=http --permanent` (enables the http service in the default zone) `# firewall-cmd --add-port=8080/tcp --permanent` (enables TCP port 8080 in the default zone) `# firewall-cmd --zone=internal --add-source=192.168.0.0/24 --permanent` (routes all traffic coming from the 192.168.0.0/24 network to the internal zone) `# firewall-cmd --zone=internal --list-all --permanent` (prints detailed information about the internal zone) `# firewall-cmd --add-rich-rule='rule family=ipv4 source address=183.131.80.130 reject' --permanent` (blocks all traffic from the specified IP address in the default zone) `# firewall-cmd --add-rich-rule='rule family=ipv4 source address=192.168.0.15 port port=8080 protocol=tcp accept' --permanent` (enables port 8080 for the specified IP address in the default zone) `# firewall-cmd --permanent --zone=work --add-rich-rule='rule family=ipv4 source address=192.168.0.0/26 forward-port port=80 protocol=tcp to-port=8080'` (forwards 80/TCP to port 8080/TCP for the specified network in the work zone) `# firewall-cmd --reload` (reloads the changes in the firewall configuration)
ssh [[<user>@]<host>] [<command>]	establishes an encrypted login to an existing account on the remote host or executes a specified command on the remote host instead of starting an interactive login shell whose output is displayed on the terminal of the local computer, **-l** <user> logs in as a specified user, **-i** <file> specifies the file with the private key (*~/.ssh/id_rsa* by default), **-p** <port> specifies a non-standard port, **-o** <option> specifies an option to override the default configuration, **-J** <jump_host> specifies a jump host, **-X** enables X11 forwarding, **-v** detailed output `$ ssh 192.168.0.20` (logs on to the remote host using the same user account on both systems) `$ ssh norton@prompt.cz` | `$ ssh -l norton prompt.cz` (logs on to the remote host using different user accounts on both systems) `$ ssh -o PubkeyAuthentication=no norton@192.168.0.20` (disables the use of the ssh key when logging in and prompts the user for a password) `$ ssh -J norton@192.168.15.107 kuba@192.168.124.5` (logs on to the remote host via a jump host using different user accounts on both systems) `$ ssh 192.168.0.20 "uname -a; who -b"` (executes the specified commands on the remote host whose output is displayed on the terminal of the local computer) `$ for server in centos{1..2}.example.com; do ssh $server 'bash -s' < script.sh; done` (executes a local script on the remote hosts)

NETWORK & COMMUNICATION	
ssh-keygen	generates a pair of authentication keys – private and public, which are used to securely identify the user during an SSH connection without having to enter their username and password; the private key is stored in *~/.ssh/id_rsa* by default, the public key is stored in *~/.ssh/id_rsa.pub* and its contents need to be put into *~/.ssh/authorized_keys* of the remote host; the program also prompts the user to enter an authentication passphrase (a string of arbitrary characters, including spaces, used to protect the private key against abuse) which, if not empty, is required for identification at the beginning of each connection, **-b** <bits> specifies the number of bits in the key (3072 for rsa by default), **-C** <comment> adds a comment, **-t** <key> specifies the type of key – "rsa", "dsa", "ecdsa" or "ed25519" ("rsa" by default), **-f** <file> specifies the file with the keys (*~/.ssh/id_rsa* and *~/.ssh/id_rsa.pub* by default), **-l** displays the key length in bits and the encryption algorithm, **-p** changes an authentication passphrase, **-v** detailed output `$ ssh-keygen -lf ~/.ssh/id_rsa | awk '{print $1}'` (prints the length of the private ssh key in bits)
ssh-copy-id [[<user>@]<host>]	copies the user's public SSH key from the local computer to *~/.ssh/authorized_keys* on the remote host, **-i** <file> specifies the file with the public key (*~/.ssh/id_rsa.pub* by default) `$ ssh-copy-id -i ~/.ssh/id_dsa.pub 94.112.152.47` (copies the public SSH key of the logged-in user to the remote computer)
ssh-agent [<command>]	allows secure logins based on SSH keys without having to enter an authentication passphrase before each connection (especially useful when remotely executing commands on a large number of hosts using a script); the "ssh-agent" is therefore executed before the start of the operation, the "ssh-add" command passes the private key to it and the authentication passphrase is entered only once `$ ssh-agent sh <-'` `$ ssh-add <-'` `> <passphrase> <-'`
ssh-add [<file>]	temporarily passes a private SSH key and authentication passphrase to the "ssh-agent" program (*~/.ssh/id_rsa*, *~/.ssh/id_dsa*, *~/.ssh/id_ecdsa* and *~/.ssh/id_ed25519* by default)
scp [[<user>@]<host>:]<source ...> [[<user>@]<host>:]<target>	establishes an encrypted data transfer between remote hosts over an SSH connection, **-i** <file> specifies the file with the private key (*~/.ssh/id_rsa* by default), **-P** <port> specifies a non-standard port, **-p** preserves file attributes, **-r** recursively, **-l** limits the used bandwidth specified in kB/s, **-C** uses compression, **-v** detailed output `$ scp ~/.ssh/id_rsa.pub 192.168.0.20:.ssh/authorized_keys` (copies the contents of the "id_rsa.pub" file from the local computer to the "authorized_keys" file on the remote host) `$ scp -rv 192.168.0.20:/home/kuba/data .` (copies the "data" directory from the remote host to the working directory on the local computer) `$ scp kuba@192.168.0.20:soubor.txt 192.168.0.21:` (copies the "file.txt" file from the user's home directory on one remote host to the user's home directory on another remote host)
sftp [[<user>@]<host>]	establishes an interactive encrypted data transfer between remote hosts over an SSH connection, **-i** <file> specifies the file with the private key (*~/.ssh/id_rsa* by default), **-P** <port> specifies a non-standard port; interactive commands: **pwd** prints the path to the working directory, **cd** <directory> changes the current directory to a specified directory, **ls** lists the contents of the working directory, **get** <file> copies a remote file to the local computer, **put** <file> copies a local file to the remote host, **!** <command> executes a specified command on the local computer, **help** | **?** help, **bye** | **quit** | **exit** quits the program
ftp [<host>]	establishes an interactive unencrypted data transfer between remote hosts; interactive commands: **pwd** prints the path to the working directory, **cd** <directory> changes the current directory to a specified directory, **ls** lists the contents of the working directory, **get** <file> copies a remote file to the local computer, **mget** <fil*> copies more remote files using wildcards, **put** <file> copies a local file to the remote host, **mput** <fil*> copies more local files using wildcards, **!** <command> executes a specified command on the local computer, **help** | **?** help, **bye** | **quit** | **exit** quits the program
telnet [<host>] [<port>]	establishes an unencrypted login to an existing account on the remote host or detects a specified port availability; with no argument it starts in interactive mode `$ telnet 192.168.0.20 80` (detects the availability of port 80 on the remote host)
lynx [<URL>]	displays the text-based contents of a URL with the ability to navigate links and interact with forms, **q** quits the program `$ lynx prompt.cz` (displays the contents of the web page)

NETWORK & COMMUNICATION	
curl [<URL ...>]	displays the source code of a URL or copies data from or to the specified URL, **-o** <file> specifies a target file (STDOUT by default), **-O** downloads the contents of a URL into the file in the working directory named after the last part of the URL path (after the trailing slash), **-F** <item>=<contents> specifies outgoing data ("@" indicates a source file), **-u** <user>:<password> specifies a username and password to use for authentication, **-x** [<protocol>://]<host>[:<port>] specifies a proxy server, **-v** detailed output `$ curl https://prompt.cz` (displays the source code of the web page) `$ curl -o script https://prompt.cz/_media/wiki/lnx.sh` (downloads the contents of the "lnx.sh" script from the website into the "script" file which is created at the same time) `$ curl -O https://prompt.cz/_media/wiki/lnx.sh` (downloads the "lnx.sh" script from the website to the working directory) `$ curl ipinfo.io/ip` (displays the public IP address of the host)
wget [<URL ...>]	downloads the contents of a URL to the working directory, **-P** <directory> specifies the download directory, **-c** continues downloading a partially downloaded file after the transfer has been interrupted, **-r** recursively, **-t** <n> specifies the number of download attempts `$ wget https://prompt.cz/_media/wiki/lnx.sh` (downloads the "lnx.sh" script from the website to the working directory)
mail	displays the contents of the logged-in user's mailbox (*/var/spool/mail/<user>*), **-f** displays the contents of the mailbox with already read messages (*/home/<user>/mbox*); interactive commands: **p** | **p**<n> displays the latest message or the specified message, **r** replies to the message, **d** | **d**<m-n> | **d*** deletes the current, specified or all messages, **q** quits the program
mail <address>	sends a message to the recipient's address, **-r** <address> specifies the sender's address, **-s** <subject> specifies the subject, **-c** <address> specifies the carbon copy (CC) address, **-b** <address> specifies the blind carbon copy (BCC) address `$ mail root < info.txt` `$ cat file | mail -s "offer" james -c root` `$ echo "Hello James" | mail -s "greeting" james` the message can also be sent this way: `$ mail <address> <-'` *`Subject:`* `<subject> <-'` `<text> <-'` `.<-'` or **Ctrl**+**d** *`Cc:`* `<address> <-'`
wall [<message>]	sends a message to the terminals of all logged-in users on the same host
write [<user>] [<terminal>]	sends a message to the specified user on the same host; if the user is logged on to multiple terminals at the same time, the terminal can be specified, otherwise the program chooses the terminal on which the user was last active `$ write kuba <-'` `<text>` **Ctrl**+**d** `$ echo "Hello" | write kuba`
talk <user>[@<host>] [<terminal>]	allows real-time communication between two users on the same host or on different hosts if they use the same usernames on both systems; if the user is logged on to multiple terminals at the same time, the terminal can be specified, otherwise the program chooses the terminal on which the user was last active `$ talk tom@prompt.cz <-'` `<text> <-'` **Ctrl**+**c**
mesg [**y** | **n**]	prints or sets the availability of the logged-in user's terminal to receive "wall", "talk" or "write" program messages ("y" = yes, "n" = no)
who -w	prints the availability of logged-in users' terminals to receive "wall", "talk" or "write" program messages ("+" = yes, "-" = no)

From:
https://prompt.cz/en/ - **Prompt.cz**

Permanent link:
https://prompt.cz/en/network-and-communication

Last update: **2025/04/24 15:57**