THE

ACTUALLY USEFUL

DEVOPS CHEAT SHEET

There are a lot of DevOps cheat sheets out there, but most of them probably include commands you already know.

This cheat sheet is our handy reference for Linux and network troubleshooting tasks.

Jump to ...

Ubuntu and Debian-specific commands

These are commands specific to Ubuntu and Debian-related distributions.

```
# list installed packages
dpkg -l
# find if a package is installed
dpkg -l | grep packagename
```

```
# find what package provides a specific file
sudo apt install apt-file
sudo apt-file update
sudo apt-file search filename
```

```
# show firewall rules
sudo ufw status
```

RHEL and CentOS-specific commands

“dnf” commands can be replaced with “yum” on RHEL/CentOS 7.

```
# list installed packages
rpm -qa
# find if a package is installed
rpm -qa | grep packagename
```

```
# find what package provides a specific file
sudo dnf provides */filename
```

```
# temporarily disable selinux
# (to see if selinux was preventing something from working)
sudo setenforce 0
# re-enable selinux
sudo setenforce 1
```

```
# show firewall rules in current zone
sudo firewall-cmd --list-all
# show all firewall rules in all zones
sudo firewall-cmd --list-all-zones
```

Network troubleshooting

These examples debug networking issues with the hypothetical machine at IP address 1.2.3.4 on port 8080.

```
# see if another machine is responsive
# (requires ICMP to be enabled in the firewall)
ping 1.2.3.4
```

```
# test connections to another machine over a specific port
nc -vz 1.2.3.4 8080
```

```
# trace the route to a machine over TCP
# (many firewalls block UDP by default)
sudo traceroute --tcp --port=8080 1.2.3.4
# get the route to a specific ip
ip route get 1.2.3.4
# print all routes
route
```

```
# get ip addresses for a machine
ip a
```

```
# listen on a specific IP address and port and print output to stdout
nc -l 127.0.0.1 8080
# send text to a service listening on a specific port and ip
# (try this out with last nc command)
telnet 127.0.0.1 8080
```

There's more....

```
# print out all tcp traffic matching a specific pattern
# (ip address, port, etc.)
sudo tcpdump -i any -nn | grep PATTERN
```

```
# get all processes listening on tcp ports
sudo ss -tlpn
# get all processes connected to a port
# (including from other machines)
sudo ss -tapn | grep 8080
```

```
# make http requests for troubleshooting
# get request, follow redirects
curl -sL 1.2.3.4:8080
# get request, headers only, and print status code of the
request
curl -sLI 1.2.3.4:8080
# get request, print all details (including certificate)
curl -sv 1.2.3.4:8080
# get request with specific host header
curl -H "Host: example.com" 1.2.3.4:8080
# post request with some json data
curl -XPOST -H "Content-Type: application/json" -d '{"data":
1}' 1.2.3.4:8080
```

```
# get all the information about a TLS certificate
openssl x509 -in some_cert.pem -noout -text
```

General Linux troubleshooting commands

```
# list all running systemd services
systemctl status
# list any failed services
systemctl list-units --state=failed
# get status of a service (mariadb for example)
systemctl status mariadb
# start/stop/restart a service
sudo systemctl start mariadb
# start a process on boot
sudo systemctl enable mariadb
```

```
# view the logs for a service as they happen
sudo journalctl -f -u mariadb
# logs since boot
sudo journalctl -b
# kernel logs
dmesg
```

```
# show all processes in realtime (use "atop" for historical
data)
top
# print process tree
ps auxwf
# find processes matching keywords
ps -ef | grep keywords
# kill a process with extreme prejudice
sudo kill -9 pid
# send specific signal to a process (SIGHUP, for example)
sudo kill -SIGHUP pid
# kill all processes matching a keyword
sudo kill -9 $(ps -ef | grep keywords | awk '{print $2}')
```

There's more....

```
# find processes that have a specific file opened
sudo lsof filename.txt
# find any deleted file descriptors still open
sudo lsof | grep deleted
# truncate a file in-place
# (without messing up processes still using it)
echo > filename.txt
```

```
# find all files with a specific extension in current directory
find . -name '*.txt'
# delete all files older than a certain number of days
find . -mtime +14 -delete
# find files with a pattern in them
grep -rin PATTERN .
```

Making changes to machines via Ansible

Savvy readers have probably noticed that none of the other sections really do anything to configure or manage machines. Making changes to hosts should always be done using a config management tool like Ansible. Here are a few examples to get you started.

```
# sample ansible hosts file
[group]
host1 ansible_host=1.2.3.4
host2 ansible_host=2.3.4.5
```

```
# verify that ansible can connect to hosts in a file
ansible -i hosts -m ping all
# restart a service on a group of hosts
ansible -i hosts -m service -a "name=mariadb state=restarted"
# execute an arbitrary command on a group of hosts
ansible -i hosts -m command -a "hostname -f"
```

```
# install ansible role (in this case, a decent mysql role)
ansible-galaxy install geerlingguy.mysql
```

```
# run ansible playbook against hosts
ansible-playbook -i hosts playbook.yml
```

```
# sample ansible playbook that installs mysql on all hosts
- hosts: all
  become: true
  roles:
    - role: geerlingguy.mysql
  tasks:
    - name: Example task that prints a message
      debug:
        msg: "See
https://docs.ansible.com/ansible/latest/collections/index_modu
le.html for a list of all the things ansible can do!"
```

Stay Connected