SSH Cheat Sheet

Advanced commands

BlowStack

SSH Tunneling - Prerequisites

enable TCP forwarding in deamon config file - /etc/ssh/sshd_config

```
AllowTcpForwarding yes
```

reload ssh servive to apply configuration changes

```
$ service ssh reload
```

SSH Tunneling - Local Port Forwarding

expose a remote server resource to a local machine

```
$ ssh -L local_port:remote_address:remote_port user@ssh_server
```

example: expose remote MySQL databse

```
$ ssh -L 5000:remote_db_server:3306 user@ssh_server
```

example: access the exposed MySQL

```
$ mysql -P 5000 -h 127.0.0.1 -u username -p
```

SSH Tunneling - Remote Port Forwarding

make local resources accessible through a remote server

```
$ ssh -R remote_port:local_address:local_port user@ssh_server
```

example: expose a local webserver for remote access

```
$ ssh -R 8080:localhost:80 user@ssh_server
```

SSH Tunneling - Dynamic Port Forwarding

expose all ports and services of a remote server through a SOCKS proxy to a local port

```
$ ssh -D local_port user@ssh_server
```

example: expose remote server resources on port 1090 (set up SOCKS proxy on local port)

```
$ ssh -D 1090 user@ssh_server
```

example: forward curl traffic from a local port through a remote server

```
$ curl --socks5 localhost:1090 https://blowstack.com
```

SSH Chaining - ProxyJump Ad hoc

access server 2 through server 1 from a local host (one jump)

```
$ ssh -J user@server1 user@server2
```

access server 3 through servers 2 and 1 from a local host (multiple jumps)

```
$ ssh -J user@server1,user@server2 user@server3
```

SSH Chaining - ProxyJump Fixed

find and use the ssh config file

```
vim ~/.ssh/config
```

access server 2 through server 1 from a local machine (one jump)

```
# Jump host
Host server1
   HostName server1@example.com
   User user1

# Final destination, using server1 as the proxy
Host server2
   HostName server2@example.com
   User user2
   ProxyJump user1@server1
```

connect to the final destination

```
$ ssh server2
```

SSH Piping

execute a command on a remote server and see results locally

```
$ ssh user@remote_server server_command | local_command
```

example: count files on a remote server

```
$ ssh user@remote_server 'ls /path/to/directory' | wc -l
```

example: retrieve and process the file locally then send the processed data to another server

```
$ ssh user@server1 'cat /path/to/remote/file' | local_processing_command | ssh user@server2 'cat > /path/to/destination/file'
```