SSH

Encrypt Key

```
openssl rsa -des3 -in key.pem -out encrypted-key.pem
# Enter a passphrase
mv encrypted-key.pem key.pem
chmod 400 key.pem
```

Decrypt Key

```
openssl rsa -in key.pem -out key.open.pem
# Enter the passphrase used to encrypt the ke
mv key.open.pem key.pem
```

Resource: https://security.stackexchange.com/questions/59136/can-i-add-a-password-to-an-existing-private-key

SSH to a host through another

You can run this command:

```
ssh -J name_of_host_to_jump_through remote_user@remote_host_ip
```

If you want to do it using `~/.ssh/config`:

```
Host remote-host
  HostName remote_host_ip
  User remote_user
  ProxyJump name_of_host_to_jump_through
```

Resources:

- had the ssh -J command
- had the config example

- got us to the config example that worked

SSH Config with jumphost

```
Host jumphost
  HostName jumphost_ip
  User jumphost_user
  # Optional if you are using private keys for auth -
  # this key needs to be on the system you're starting from:
  IdentityFile ~/.ssh/jumphost-ssh-key.pem

Host targethost
  HostName targethost_ip
  User targethost_user
  ProxyJump jumphost
  # Optional if you are using private keys for auth -
  # this key also needs to be on the system you're starting from -
  # it will not work if it's on the jumphost:
  IdentityFile ~/.ssh/targethost-ssh-key.pem
```

Resource: https://serverfault.com/questions/337274/ssh-from-a-through-b-to-c-using-private-key-on-b

Specify directory for when you ssh in

Simply add the following to your `~/.bashrc` or `~/.zshrc`, etc:

```
cd /dir/to/start/in
```

Proper SSH Permissions

```
# Directory - 700
chmod 700 ~/.ssh

# Private keys - 600
chmod 600 ~/.ssh/id_rsa

# Config file - 600
```

```
chmod 600 ~/.ssh/config

# Public keys - 600 or 644
chmod 600 ~/.ssh/id_rsa.pub

# Make public key readable by others
chmod 644 ~/.ssh/id_rsa.pub

# Make authorized_keys readable by others.
# Should be 600 or 644
chmod 644 ~/.ssh/authorized_keys
```

Resource: https://unix.stackexchange.com/questions/257590/ssh-key-permissions-chmod-settings

Status of SSH Service

```
sudo systemctl status sshd.service
```

Restart SSH Service

```
sudo systemctl restart sshd.service
```

Resource: https://www.cyberciti.biz/faq/centos-stop-start-restart-sshd-command/##centos_7_centos_8

Forward local service to remote host

This will forward a service running on localhost:3000 to a remote host on port 3000, and will allow other systems on the network to access that service.

On the remote host, run this command to add a line to the `sshd_config`:

```
echo 'GatewayPorts clientspecified' | sudo tee -a /etc/ssh/sshd_config
```

Next, restart the ssh service:

```
service ssh restart
```

Finally, run this command from the system running the service:

```
ssh -R :3000:localhost:3000 user@$target_server
```

Resources: https://serverfault.com/questions/861909/ssh-r-make-target-host-accept-connection-on-all-interfaces https://serverfault.com/questions/33283/how-to-setup-ssh-tunnel-to-forward-ssh

Run command over SSH

```
ssh ubuntu@yoursystem 'sudo apt-get update'
```

Resource: https://www.cyberciti.biz/faq/unix-linux-execute-command-using-ssh/

Run multiple commands with ssh

```
ssh -i ${KEY} ${USER}@${IP} << EOF
export AWS_ACCESS_KEY_ID=${ACCESS_KEY}
export AWS_SECRET_ACCESS_KEY=${SECRET_ACCESS_KEY}
export AWS_SESSION_TOKEN=${SESSION_TOKEN}
curl -sLO "https://dl.k8s.io/release/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
chmod +x kubectl
aws eks --region ${AWS_REGION} update-kubeconfig --name ${TARGET_CLUSTER}
./kubectl get pods -n ${TARGET_NAMESPACE}
EOF
}
```

Resource: https://www.shellhacks.com/ssh-execute-remote-command-script-linux/

Forward remote service to localhost

This example will forward the service running on `target.server:8080` to `localhost:8080`:

```
ssh -L 8080:localhost:8080 username@target.server
```

Forward remote service via bastion to localhost

This example will forward the service running on `target.server:443` to `localhost:4455`:

```
ssh -N -L 4455:target.server:443 username@bastion
```

Alternatively if you want to use a SOCKS proxy (say you're using burp for example), you can do the following:

```
ssh -C -D 8085 username@bastion
```

and then point your SOCKS proxy to localhost:8085 in order to hit hosts only accessible via the bastion.

Resources: http://www.spencerstirling.com/computergeek/sshtunnel.html

https://medium.com/@mccabe615/proxying-burp-traffic-e6e7a8adc101

View images on a remote system

```
ssh -Y user@server
apt install -y eog
eog pictures/foo.png
```

Resource: https://superuser.com/questions/557622/how-can-i-view-pictures-via-ssh

Transfer file to host running SSH on a particular port

```
PORT=2222
scp -P $PORT file user@system
```

Resource: https://askubuntu.com/questions/182478/ssh-scp-to-copy-file-to-remote-server-port-21

Transfer file on behalf of another system

This is useful if you have two aws instances, and want to transfer a file between them from your laptop.

```
scp -3 -i $PEM_FILE user1@system_with_file:/file/to/xfer
user2@system_that_needs_file:/file/to/xfer
```

Resource: https://superuser.com/questions/686394/scp-between-two-remote-hosts-from-my-third-pc

Fix SSH Too Many Authentication Failures error

Add this line to your `~/.ssh/config`:

```
Host *
  IdentitiesOnly=yes
```

Resource: https://www.tecmint.com/fix-ssh-too-many-authentication-failures-error/

SSH master mode

Quick test with password:

```
sshpass -p 'password123' ssh -o StrictHostKeyChecking=no -N -M -S /tmp/mysock \
  ubuntu@192.168.1.2 &
sleep 2
ssh -S /tmp/mysock ubuntu@192.168.1.2 exit
sleep 2
ssh -S /tmp/mysock -O exit ubuntu@192.168.1.2
```

Resource: https://unix.stackexchange.com/questions/32984/multiple-ssh-sessions-in-single-command

SCP files using wildcard

Be sure to escape the wildcard, i.e. `file-\*`

Full example:

```
scp ubuntu@target:~/.config/cred\*
```

Resource: https://unix.stackexchange.com/questions/27419/how-to-use-wildcards-when-copying-with-scp

Check if you can ssh to several hosts

`hosts.txt`:

```
host1.com
192.168.1.2
```

`test_ssh.sh`:

```
for SERVER in $(cat hosts.txt); do
  ssh -i id_rsa -o StrictHostKeyChecking=no -o BatchMode=yes \
    user@$SERVER exit && echo OK $SERVER || echo ERR $SERVER
done
```

Resource: https://stackoverflow.com/questions/49564299/script-to-check-if-i-can-access-multiple-servers

Create SSH key script

```
KEY_NAME="custom"

if [ ! -e ~/.ssh/$KEY_NAME.pub ]; then
  echo
  echo 'Creating your public and private ssh keys'
  echo '---------------------------------------'

  # Create public and private ssh key pair without pw prompt
  ssh-keygen -t ed25519 -C "Key Description" -f "~/.ssh/${KEY_NAME}" -N ''
  # RSA if you prefer
  #ssh-keygen -t rsa -C "Key Description" -f "~/.ssh/${KEY_NAME}" -N ''
  # PEM file example
  #ssh-keygen -t rsa -m PEM -C "Key Description" -f "~/.ssh/${KEY_NAME}" -N ''
  echo

  # Copy new key to make it possible to autologin
  cat "~/.ssh/${KEY_NAME}.pub" >> ~/.ssh/authorized_keys
  chmod 0600 ~/.ssh/authorized_keys

  # Add the ssh key
  eval "$(ssh-agent)"
  ssh-add ~/.ssh/$KEY_NAME
else
  echo
```

```
  echo "Public ssh key file ${KEY_NAME} already exists"
  echo "----------------------------------------"
fi
```

Resources: https://stackoverflow.com/questions/10767488/automate-ssh-keygen-t-rsa-so-it-does-not-ask-for-a-passphrase https://unix.stackexchange.com/questions/48863/ssh-add-complains-could-not-open-a-connection-to-your-authentication-agent/48868

Generate public key from private key

```
ssh-keygen -y -f ~/.ssh/id_rsa > ~/.ssh/id_rsa.pub
```

Resource: https://askubuntu.com/questions/53553/how-do-i-retrieve-the-public-key-from-a-ssh-private-key

Show successful ssh logins

```
grep sshd.\*Accepted /var/log/auth.log
```

Show failed ssh logins

```
grep sshd.\*Failed /var/log/auth.log
```

SSHD with custom file

`sshd -f <path/to/sshd_config/file>` For example: `sshd -f /tmp/sshd_config`

The sshd_config file can look roughly like this:

```
Port 6022
HostKey /tmp/sshd_config/host_rsa
PasswordAuthentication yes
PermitRootLogin no
```

Fix known_hosts error

You should not use this unless you are dealing with a test host that you created.

```
TARGET_HOST=some-system
ssh-keygen -R $TARGET_HOST
```

Resource: https://kinsta.com/knowledgebase/warning-remote-host-identification-has-changed/

Revision #1
Created 2023-11-10 06:12:02 UTC by Ryan
Updated 2025-02-12 01:10:55 UTC by Ryan