

**PSI**

Center for Scientific Computing,  
Theory and Data

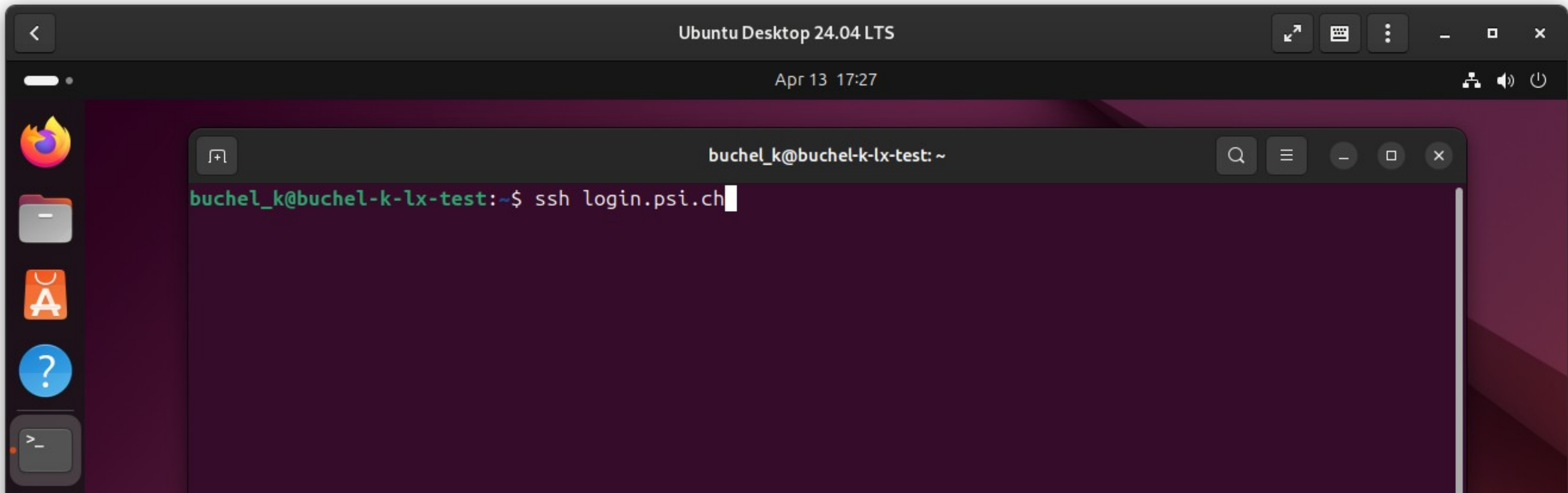
# SSH Deep Dive

Konrad Bucheli <[konrad.bucheli@psi.ch](mailto:konrad.bucheli@psi.ch)>  
RSE Seminar, 5 May 2026

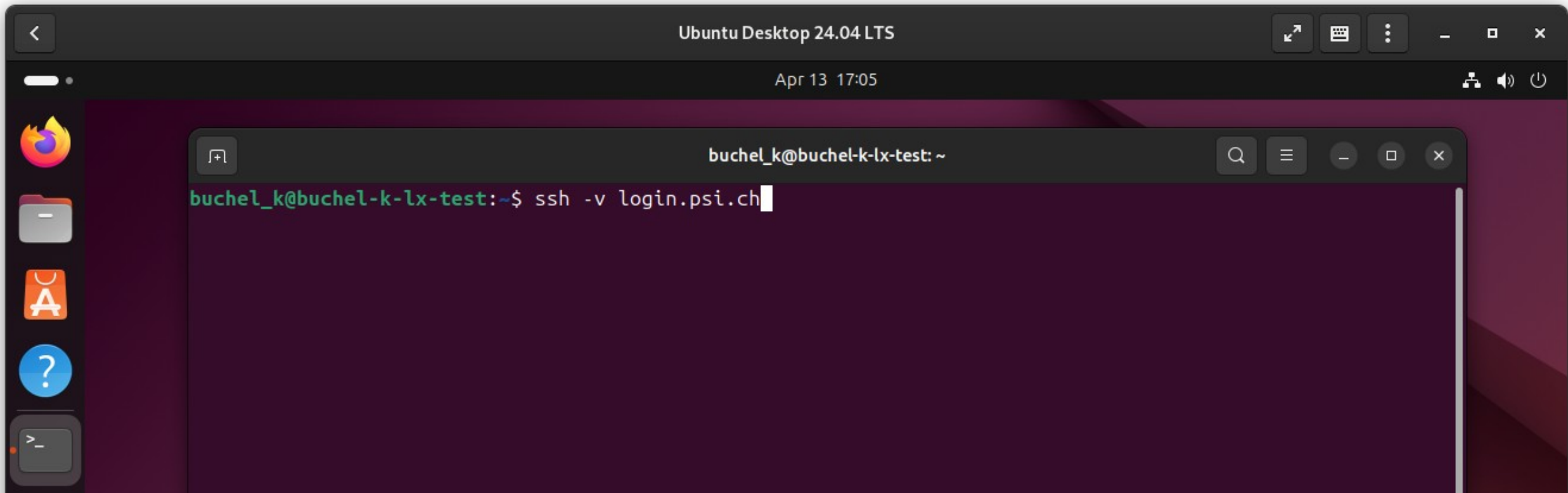
## Agenda

- **Connecting to a Machine (`ssh -v`)**
- **Authentication**
- **Tunneling**
- **Multiplexing with Master Mode**
- **Client Configuration**
- **SSH Certificates**

# Connecting to a Machine



# ssh -v



# ssh -v



## what we use:

OpenSSH\_9.6p1 Ubuntu-3ubuntu13.14, OpenSSL 3.0.13 30 Jan 2024

## client configuration:

```
debug1: Reading configuration data /etc/ssh/ssh_config
debug1: /etc/ssh/ssh_config line 19: include /etc/ssh/ssh_config.d/*.conf matched no files
debug1: /etc/ssh/ssh_config line 21: Applying options for *
```

## TCP connection:

```
debug1: Connecting to login.psi.ch [129.129.190.46] port 22.
debug1: Connection established.
```



# ssh -v



## any private keys on my side?

```
debug1: identity file /home/buchel_k/.ssh/id_rsa type -1
debug1: identity file /home/buchel_k/.ssh/id_rsa-cert type -1
...
debug1: identity file /home/buchel_k/.ssh/id_dsa type -1
debug1: identity file /home/buchel_k/.ssh/id_dsa-cert type -1
```

## any known compatibility issues with server?

```
debug1: Local version string SSH-2.0-OpenSSH_9.6p1 Ubuntu-3ubuntu13.14
debug1: Remote protocol version 2.0, remote software version OpenSSH_8.0
debug1: compat_banner: match: OpenSSH_8.0 pat OpenSSH* compat 0x04000000
```

## this is a bit early !?!?

```
debug1: Authenticating to login.psi.ch:22 as 'buchel_k'
debug1: load_hostkeys: fopen /home/buchel_k/.ssh/known_hosts: No such file or directory
...
```

# ssh -v



## start key exchange:

```
debug1: SSH2_MSG_KEXINIT sent
debug1: SSH2_MSG_KEXINIT received
debug1: kex: algorithm: curve25519-sha256
debug1: kex: host key algorithm: ssh-ed25519-cert-v01@openssh.com
debug1: kex: server->client cipher: chacha20-poly1305@openssh.com MAC: <implicit> compression: none
debug1: kex: client->server cipher: chacha20-poly1305@openssh.com MAC: <implicit> compression: none
debug1: expecting SSH2_MSG_KEX_ECDH_REPLY
debug1: SSH2_MSG_KEX_ECDH_REPLY received
```

## server banner:

```
debug1: SSH2_MSG_SERVICE_ACCEPT received
```

```
----
```

```
This system has local homes. Access to AFS is possible via /afs/psi.ch/...
```

```
----
```

# ssh -v



## client authenticating public key of server:

```
debug1: Server host certificate: ssh-ed25519-cert-v01@openssh.com \
      SHA256:+gNeSv9Y1Uhd7QMtcoGbrSw2K8OysUMPCAyUMBPVhXc, serial 0 ID "lx-login-01.psi.ch" \
      CA ssh-ed25519 SHA256:wESKK901P+fMzBkA+P040CGpfA72GsHb8LRVE8EdR5g \
      valid from 2026-03-16T06:04:09 to 2026-09-08T07:04:09
debug1: load_hostkeys: fopen /home/buchel_k/.ssh/known_hosts: No such file or directory
...
debug1: No matching CA found. Retry with plain key
debug1: hostkeys_find_by_key_hostfile: hostkeys file /home/buchel_k/.ssh/known_hosts does not exist
...
The authenticity of host 'login.psi.ch (129.129.190.46)' can't be established.
ED25519 key fingerprint is SHA256:+gNeSv9Y1Uhd7QMtcoGbrSw2K8OysUMPCAyUMBPVhXc.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'login.psi.ch' (ED25519) to the list of known hosts.
```



# ssh -v



## continue key exchange:

```
debug1: ssh_packet_send2_wrapped: resetting send seqnr 3
debug1: rekey out after 134217728 blocks
debug1: SSH2_MSG_NEWKEYS sent
debug1: expecting SSH2_MSG_NEWKEYS
debug1: ssh_packet_read_poll2: resetting read seqnr 3
debug1: SSH2_MSG_NEWKEYS received
debug1: rekey in after 134217728 blocks
debug1: SSH2_MSG_EXT_INFO received
debug1: kex_ext_info_client_parse: server-sig-algs=<ssh-ed25519,ssh-rsa,rsa-sha2-256,\
    rsa-sha2-512,ssh-dss,ecdsa-sha2-nistp256,ecdsa-sha2-nistp384,ecdsa-sha2-nistp521>
```

# ssh -v



## server authenticating user:

```
debug1: Authentications that can continue: publickey,gssapi-keyex,gssapi-with-mic,password
debug1: Next authentication method: gssapi-with-mic
debug1: No credentials were supplied, or the credentials were unavailable or inaccessible
No Kerberos credentials available (default cache: FILE:/tmp/krb5cc_1000)
debug1: Next authentication method: publickey
debug1: get_agent_identities: bound agent to hostkey
debug1: get_agent_identities: ssh_fetch_identitylist: agent contains no identities
debug1: Will attempt key: /home/buchel_k/.ssh/id_rsa
...
debug1: Trying private key: /home/buchel_k/.ssh/id_dsa
debug1: Next authentication method: password

buchel_k@login.psi.ch's password:
Authenticated to login.psi.ch ([129.129.190.46]:22) using "password".
```

# ssh -v



## server setting up user session:

```
debug1: channel 0: new session [client-session] (inactive timeout: 0)
debug1: Requesting no-more-sessions@openssh.com
debug1: Entering interactive session.
debug1: pledge: filesystem
debug1: client_input_global_request: rtype hostkeys-00@openssh.com want_reply 0
debug1: client_input_hostkeys: searching /home/buchel_k/.ssh/known_hosts for login.psi.ch / (none)
debug1: client_input_hostkeys: searching /home/buchel_k/.ssh/known_hosts2 for login.psi.ch / (none)
debug1: client_input_hostkeys: hostkeys file /home/buchel_k/.ssh/known_hosts2 does not exist
debug1: Sending environment.
debug1: channel 0: setting env LANG = "en_US.UTF-8"
...
debug1: pledge: fork

Last login: Mon Apr 13 14:42:57 2026 from mgt-gw-01.psi.ch
[buchel_k@lx-login-01 ~]$
```

# Authentication



debug1: Authentications that can continue: **publickey,gssapi-keyex,gssapi-with-mic,password**

## what authentication methods exist?

- **password**
- **keyboard-interactive**: same as password, but allows for challenge response (e.g. MFA TOTP)
- **none**: for password less users if allowed
- **gssapi-with-mic, gssapi-keyex**: Kerberos
- **publickey**: trusted public key
- **hostbased**: trust that selected client hosts have properly authenticated selected users

# Authentication: Kerberos

## Requirements:

- server known to KDC (joined to the ActiveDirectory (AD))
- user known to KDC (AD-user)
- user is authenticated against KDC (AD) and has a TGT (Ticket Granting Ticket)

## short:

works from managed RHEL to managed RHEL

self-managed Linux: <https://linux.pages.psi.ch/cookbook/ubuntu/kerberos/#installation>

## Identity (TGT) forwarding:

- command: `ssh -K ...`
- configuration: `GSSAPIDelegateCredentials yes`

# Authentication: Public Key

## Requirements:

- client user needs a private/public key pair
- accepted public key (or certificate) configured on server per user (`~/.ssh/authorized_keys`)

## What public key types are available in ssh?

- **rsa:** supported everywhere, but at times short keylength and/or SHA1 digests get banned
- **ed25519:** default, fast, short keys
- **ecdsa:** why not?
- **dsa:** please don't!
- **xmss:** supposed to be post quantum cryptography safe, if you manage to create such a key



# Authentication: Public Key



## Create your public/private keypair:

```
$ ssh-keygen -t ed25519 -C konrad.bucheli@psi.ch
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/buchel_k/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/buchel_k/.ssh/id_ed25519.
Your public key has been saved in /home/buchel_k/.ssh/id_ed25519.pub.
The key fingerprint is:
SHA256:7iXPUCsx8tI5IACcZ5DNVan7GOWtJkU6TxLbZW6VUNI konrad.bucheli@psi.ch
The key's randomart image is:
+--[ED25519 256]--+
|o.* ...+=..      |
|= * . +.E         |
|+ * = .           |
| * = o            |
| O = S .          |
|. O * = .         |
|+ * X o           |
|+ + O             |
|. . o             |
+-----[SHA256]-----+
```

# Authentication: Public Key



## Distribute your public key:

```
$ cat /home/buchel_k/.ssh/id_ed25519.pub  
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIAVZ9yzBT5+2fnhiUOQ8VShs9VyRQAvbPEzzbXq5szcR konrad.bucheli@psi.ch  
$
```

- append to any servers `~/.ssh/authorized_key` you want to connect passwordless
- run `ssh-copy-id` for any servers you want to connect passwordless
- to any Git Forge you work with
- to everyone interested in the world

**no Kerberos:** run `kinit`

# Authentication: Public Key

## Protect your private key!

- keep it always, always, **always passphrase protected**
- when starting work, load it to the agent:

```
$ ssh-add
Enter passphrase for /home/buchel_k/.ssh/id_ed25519:
Identity added: /home/buchel_k/.ssh/id_ed25519 (konrad.bucheli@psi.ch)
$
```

### Identity (Agent) forwarding:

- command: `ssh -A ...`
- configuration: `ForwardAgent yes`

# Authentication: Host Based



Used in trusted environments:

- between cluster members
- from a secure management station

```
[worker@compute01 ~]# cat ~/.shosts
mgmt01.psi.ch root
mgmt02.psi.ch root
[worker@compute01 ~]
```

=> allow to get user `worker` on this machine if you are `root` on `mgmt0[1,2].psi.ch`

## Requirements:

client host public key needs to be known to server

server configuration: `HostbasedAuthentication yes`

client configuration: `HostbasedAuthentication yes`  
`EnableSSHKeysign yes`

# Tunneling: GUI Windows

```
debug1: channel 0: new session [client-session] (inactive timeout: 0)
```

## X forwarding:

- command: `ssh -Y ...`
- configuration: `ForwardX11 yes`  
`ForwardX11Trusted yes`

## Wayland forwarding:

needs `waypipe` installed on client and server

- command: `waypipe ssh ...`

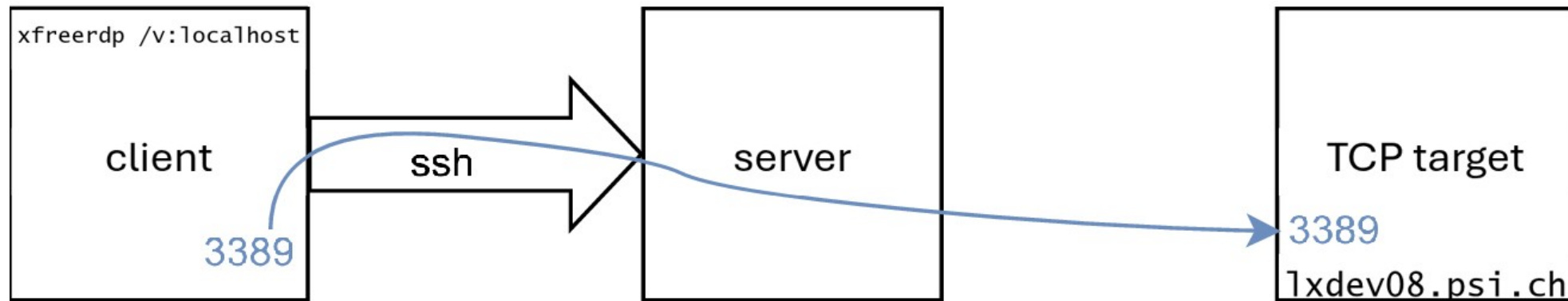
Electron based applications need some persuasion:

```
$ XDG_SESSION_TYPE=wayland code
```

# Tunneling: Traffic Forwarding

## Forwarding to remote TCP port:

- command: `ssh -L 3389:lxdev08.psi.ch:3389 ...`
- configuration: `LocalForward 3389 lxdev08.psi.ch:3389`

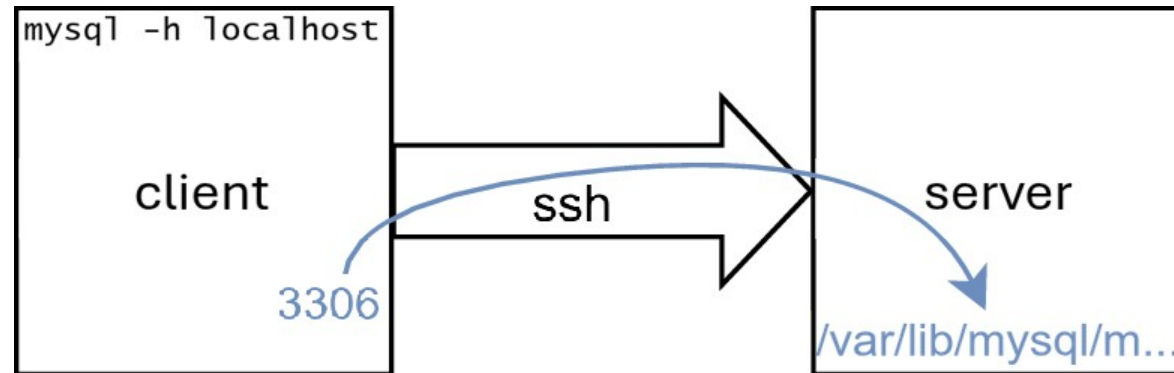




# Tunneling: Traffic Forwarding

## Forwarding to UNIX domain socket:

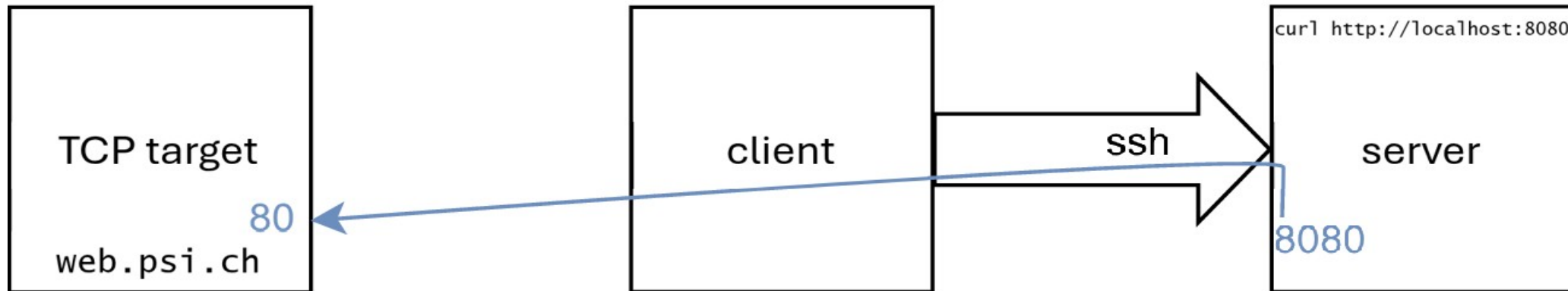
- command: `ssh -L 3306:/var/lib/mysql/mysql.sock ...`
- configuration: `LocalForward 3306 /var/lib/mysql/mysql.sock`



# Tunneling: Traffic Forwarding

## Reverse TCP port forwarding:

- command: `ssh -R 8080:web.psi.ch:80 ...`
- configuration: `RemoteForward 8080 web.psi.ch:80`



# Tunneling: Traffic Forwarding

## Source port binding:

- by default binds to `lo` interface (local loopback)
- make forwarded port available over network:  
`ssh -L *:3389:lxdev08.psi.ch:3389 ...`
- source can also be a UNIX domain socket:  
`ssh -L /run/user/45256/lxdev08_rdp:lxdev08.psi.ch:3389 ...`

*38. There are very few network restrictions creative and determined use of ssh(1) port forwarding can't overcome.*  
*39. This is both incredibly useful and concerning.*

<https://netmeister.org/blog/ops-lessons.html>

# Tunneling: Proxy Jump

Using a «SSH gateway»/«jump host»/«bastion host»:

```
buchel_k@mpc3325:~$ ssh mgt-gw.psi.ch  
Last login: Fri Apr 17 12:54:05 2026 from 129.129.175.171  
[buchel_k@mgt-gw-01 ~]$ ssh lxdev08.psi.ch  
Last login: Mon Apr 20 13:10:26 2026 from mgt-gw-01.psi.ch  
[buchel_k@lxdev08 ~]$
```

**Alternative: direct proxy jump:**

- command: `ssh -J mgt-gw.psi.ch lxdev08.psi.ch`
- configuration: `ProxyJump mgt-gw.psi.ch`

# Tunneling: SOCKS Proxy



## Forwarding:

- command: `ssh -D 9999 ...`
- configuration: `DynamicForward 9999`

set your browsers proxy:

## or proxy.pac file (Automatic proxy configuration URL):

```
function FindProxyForURL(url, host) {  
    var sock_proxy = "SOCKS5 localhost:9999; DIRECT";  
    if (dnsDomainIs(host, ".psi.ch") {  
        return sock_proxy;  
    }  
    var resolved_ip = dnsResolve(host);  
    if (  
        isInNet(resolved_ip, "129.129.0.0", "255.255.0.0")  
        || isInNet(resolved_ip, "192.33.120.0", "255.255.248.0")  
        || isInNet(resolved_ip, "192.33.118.0", "255.255.254.0")  
        || isInNet(resolved_ip, "10.0.0.0", "255.0.0.0")  
        || isInNet(resolved_ip, "172.16.0.0", "255.240.0.0")  
        || isInNet(resolved_ip, "192.168.0.0", "255.255.0.0")  
    ) {  
        return sock_proxy;  
    }  
    return "DIRECT";  
}
```

Connection Settings

Configure Proxy Access to the Internet

☐ No proxy

☐ Auto-detect proxy settings for this network

☐ Use system proxy settings

☒ Manual proxy configuration

HTTP Proxy  Port

☐ Also use this proxy for HTTPS

HTTPS Proxy  Port

SOCKS Host  Port

☐ SOCKS v4 ☒ SOCKS v5

☐ Automatic proxy configuration URL

Reload

No proxy for

Example: .mozilla.org, .net.nz, 192.168.1.0/24  
Connections to localhost, 127.0.0.1/8, and ::1 are never proxied.

☐ Do not prompt for authentication if password is saved

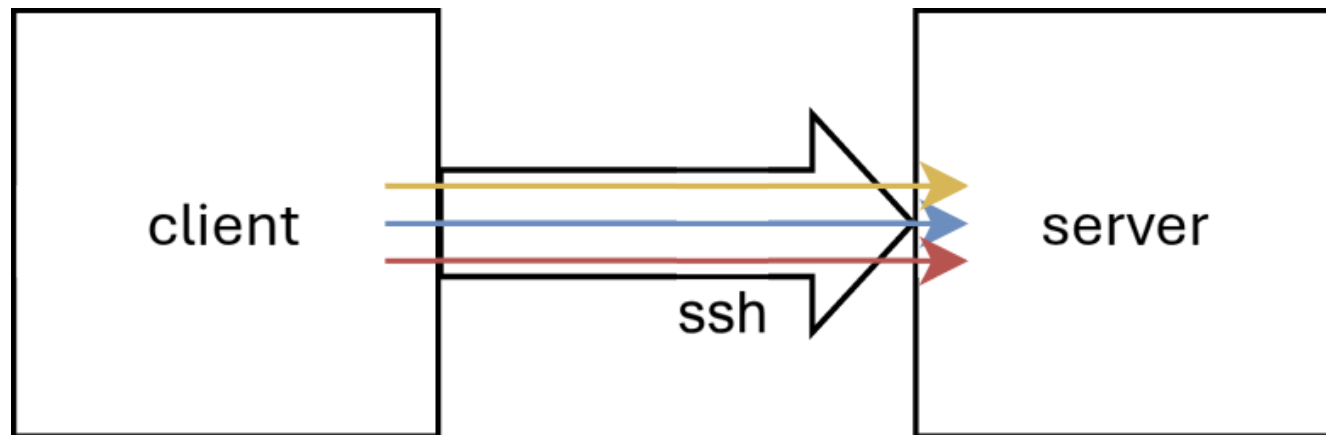
☐ Proxy DNS when using SOCKS v4

☒ Proxy DNS when using SOCKS v5

Cancel OK

# Multiplexing with Master Mode

```
debug1: channel 0: new session [client-session] (inactive timeout: 0)
```



one ControlMaster connection which handles for once

- connection setup
- authentication
- Kerberos, agent and X forwarding

to be used for activities following up



# using Master Mode Explicitly

## create control connection:

```
$ ssh -M -S my-ssh.socket -N -f lxdev08.psi.ch  
$
```

- `-M` enables master mode
- `-S` shows which UNIX domain socket to use to control it
- `-N` runs no remote command
- `-f` goes into the background

## check control connection:

```
$ ssh -S my-ssh.socket -O check lxdev08.psi.ch  
Master running (pid=3587600)  
$
```

- `-O` command for the control connection

# using Master Mode Explicitly



## use control connection for remote shell:

```
$ ssh -S my-ssh.socket -v lxdev08.psi.ch
OpenSSH_9.6p1 Ubuntu-3ubuntu13.14, OpenSSL 3.0.13 30 Jan 2024
debug1: Reading configuration data /etc/ssh/ssh_config
debug1: /etc/ssh/ssh_config line 19: include /etc/ssh/ssh_config.d/*.conf matched no files
debug1: /etc/ssh/ssh_config line 21: Applying options for *
Last login: Tue Apr 21 11:23:36 2026 from lx-login-01.psi.ch
[buchel_k@lxdev08 ~]$
```

## use control connection to add port forwarding:

```
$ ssh -S my-ssh.socket -O forward -L 3389:localhost:3389 lxdev08.psi.ch
$
```

## close control connect/exit master mode:

```
$ ssh -S my-ssh.socket -O exit lxdev08.psi.ch
Exit request sent.
$
```

# using Master Mode Implicitly

## Client configuration:

```
ControlMaster auto  
ControlPersist 3h  
ControlPath ~/.ssh/controlmaster-%r@%h:%p
```

- `ControlMaster auto`: automatically create a master connection/control socket upon first use
- `ControlMaster`: how long shall the master connection be kept open if unused
- `ControlPath`: path of control socket, here individually per remote user, host and port

=> Master Mode is automatically set up in the background and transparently used

# Client Configuration

## Goal:

```
$ ssh $TARGET-HOST
```

ideally should just work, because then all `ssh` backed tools (`rsync`, code integration, ...) will also just work

=> tailor it individually to your usage (mine has > 200 lines)

# Client Configuration (~/.ssh/config)

```
# globally enforced configuration
ControlPersist 3h
ControlPath ~/.ssh/controlmaster_%r@%h:%p
HashKnownHosts no
AddKeysToAgent yes

# host specific configuration
...

# default configuration
User buchel_k
    GSSAPIDelegateCredentials yes

Host *
    ControlMaster auto
    ServerAliveInterval 300
    ServerAliveCountMax 2
```

# Client Configuration (~/.ssh/config)



```
# git forges safety
Host gitlab.com github.com gitea.psi.ch gitea-test.psi.ch bitbucket.org
    ForwardX11 no
    ForwardX11Trusted no
    GSSAPIDelegateCredentials no
    ForwardAgent no

# hopx VPN
Host hopx hopx.psi.ch
    Hostname hopx.psi.ch
    DynamicForward 9999
Match host "mgt-gw*,gitea-test*" exec "ssh -O check hopx.psi.ch"
    ProxyJump hopx.psi.ch

# target hosts
host lx* pc*
    ProxyJump mgt-gw.psi.ch
```



# Client Configuration (~/.ssh/known\_hosts)



Stores public keys of all known hosts, like:

```
github.com ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIOMqqnkVzrm0SdG6UOoqKLsabgH5C9okWi0dh2l9GKJl
```

to accept all host keys of all Puppet managed RHEL hosts add

```
# SSH host key certificate CA managed by PSI Puppet
@cert-authority * ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBF2xLCHNmXSeY+qNPmdo/jO2AXrcHyQDqZLBzWVwk2/GqMRDl4mavZLBPYQPNELGAPc+BHg7iRC65wQQEeYOVU=
@cert-authority * ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIKhIx/obIiiO9AtrUgj7pF8kAgv4g9Dp+DRXilkmlkxK
```

# SSH Certificates

## **host public key certificates:**

- for a big fleet of machines

## **user public key certificates:**

- for a big flock of users
- for role based access
- for time limited access
- disable some features
- for IP limited access

## **disadvantage:**

- CA operation

# SSH User Certificates



```
$ ssh-keygen -L -f .ssh/id_ed25519-cert.pub
.ssh/id_ed25519-cert.pub:
    Type: ssh-ed25519-cert-v01@openssh.com user certificate
    Public key: ED25519-CERT SHA256:uUJfFx2j5I6IQGG1IwA0iPLMavYlUUNn9cOxTiAhtUI
    Signing CA: RSA SHA256:TFLrrYzzyCyewyx+LmlrzKjQPmZaLPLDB8aJSDZR9l8 (using rsa-sha2-512)
    Key ID: "buchel_k"
    Serial: 0
    Valid: from 2026-01-08T11:37:00 to 2027-01-28T11:38:02
    Principals:
        buchel_k
    Critical Options: (none)
    Extensions:
        permit-X11-forwarding
        permit-agent-forwarding
        permit-port-forwarding
        permit-pty
        permit-user-rc
```