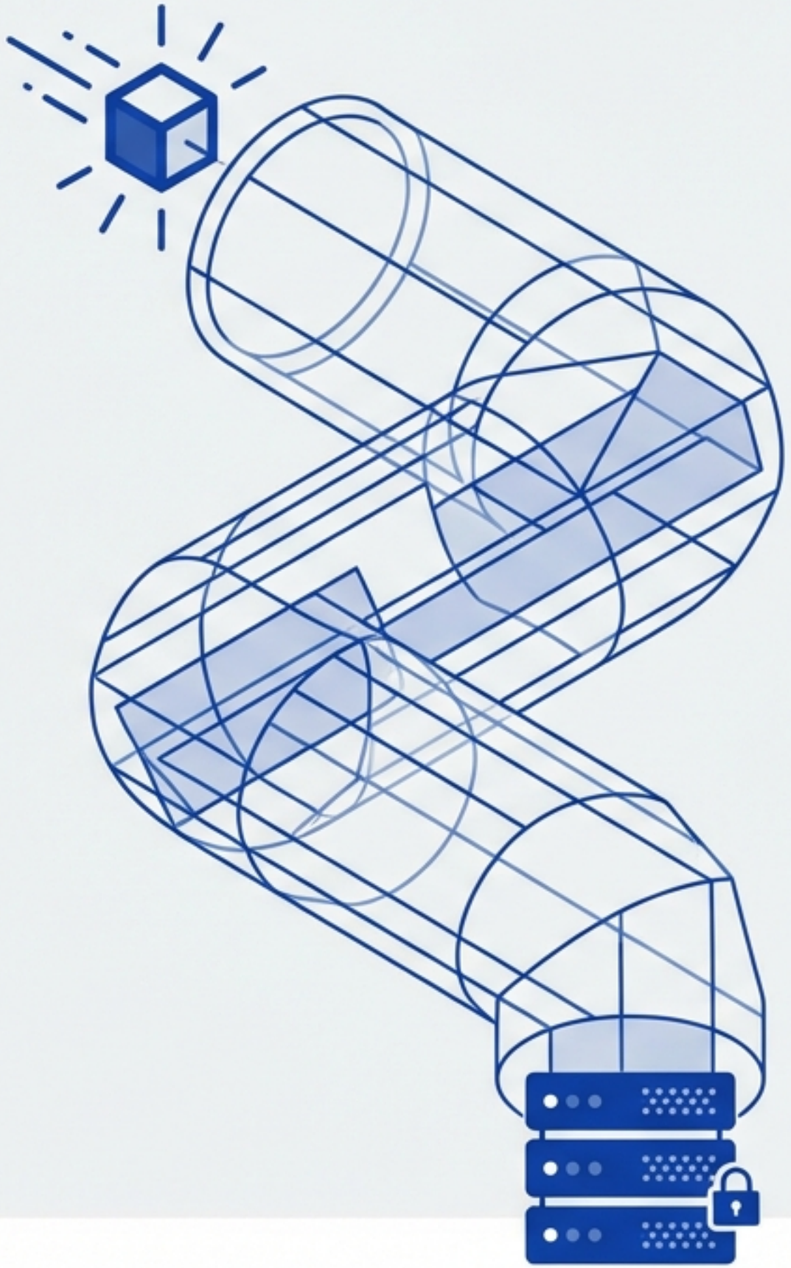
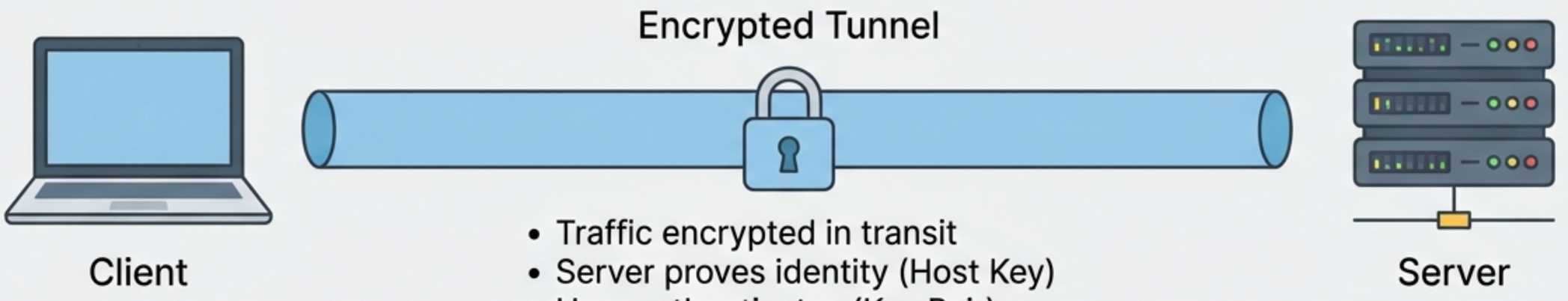
Mastering SSH for Data Science

Secure Access, File Transfer, and Automation



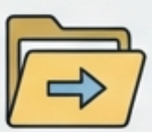
What is SSH?

SSH (Secure Shell) is the standard protocol for encrypted remote login and command execution over insecure networks. It replaces legacy passwords with cryptographic verification.




Remote Shell


Remote Execution


File Transfer


Git Auth


DB Tunneling

The Anatomy of a Key Pair

Private Key

The key in your pocket.
Stays on your local laptop. Protected by file permissions and optional passphrase.
NEVER shared.

Public Key

The lock you place on the door.
Copied to the remote server.
Safe to share.
Stored in authorized_keys.

The Mechanism:
The server uses the Public Key (lock) to create a challenge that only the Private Key can answer.

Step 1: Generating Your Keys

Perform this once per machine (laptop/workstation).

```
# Generate an Ed25519 key (modern, secure)
ssh-keygen -t ed25519 -C 'your.name@yourorg.edu'
```

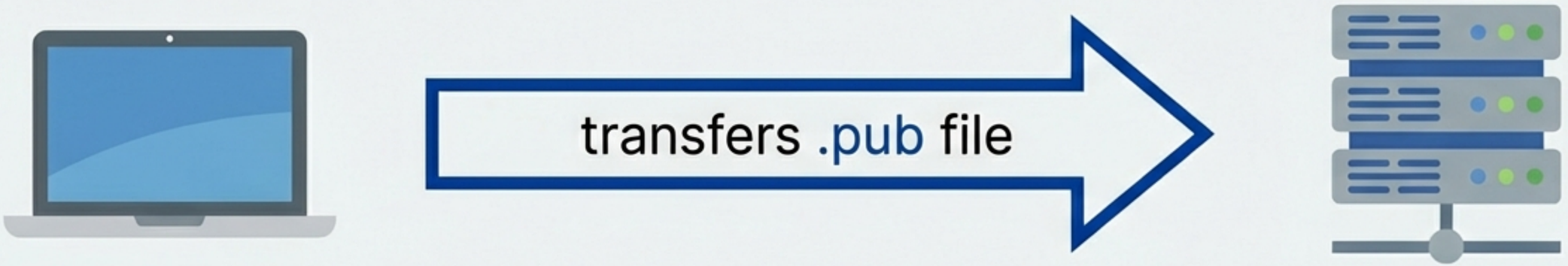
~/.ssh/id_ed25519
(Private)

~/.ssh/id_ed25519.pub
(Public)

During generation, press Enter to accept the default file path. Set a passphrase for added security, or press Enter for automation (no passphrase).

Step 2: Installing Keys (The Easy Way)

Use ssh-copy-id to automatically transfer your public identity.




```
# Uses password once to install the key
ssh-copy-id -i ~/.ssh/id_ed25519.pub youruserid@remote.host.edu
```

This command logs into the server, creates the `~/.ssh` directory if needed, and appends your public key to the `authorized_keys` file with the correct permissions. Subsequent logins will rely on the key, not the account password.

Step 2 (Alternative): Manual Key Installation

For Windows users or when ssh-copy-id is unavailable.

Local Machine

Copy public key text:

```
cat ~/.ssh/id_ed25519.pub
```

~/.ssh/id_ed25519.pub
(Public)

Remote Connection

Login with password:

```
ssh youruserid@remote.host.edu
```

Remote Server

Paste into authorized_keys:

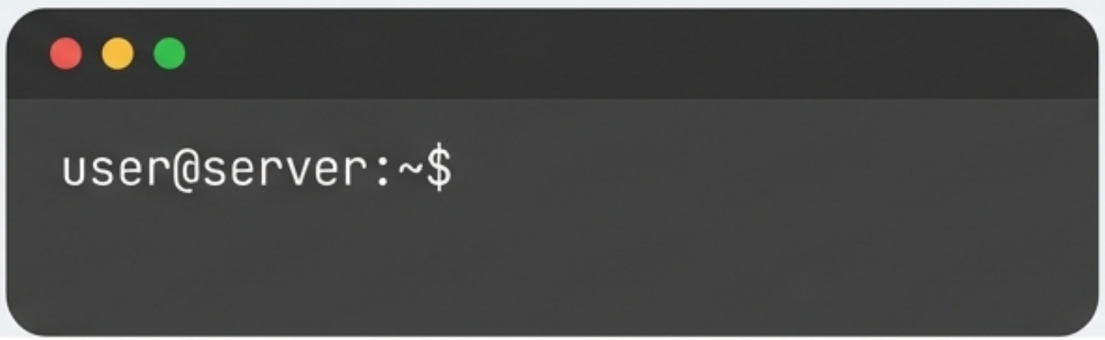
```
mkdir -p ~/.ssh
chmod 700 ~/.ssh
nano ~/.ssh/authorized_keys
# Paste key, save, exit
chmod 600 ~/.ssh/authorized_keys
```

Remote Shell & Command Execution

Interactive Shell

Full login session to work on the server.

```
ssh youruserid@remote.host.edu
```



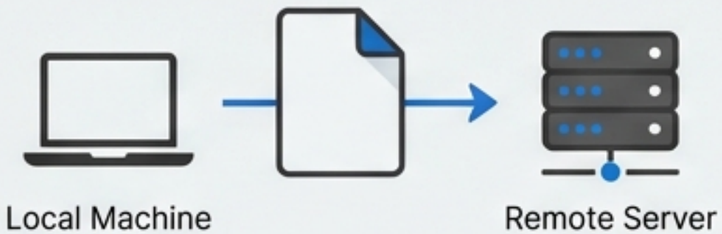
Single Command

Trigger a script without a full login session.

```
ssh youruserid@remote.host.edu
'python train_model.py --epochs 10'
```

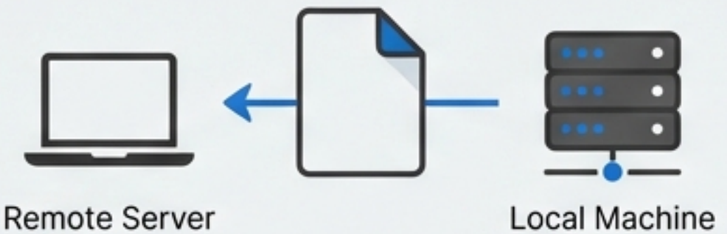
Moving Files with SCP

Securely transferring datasets and results.



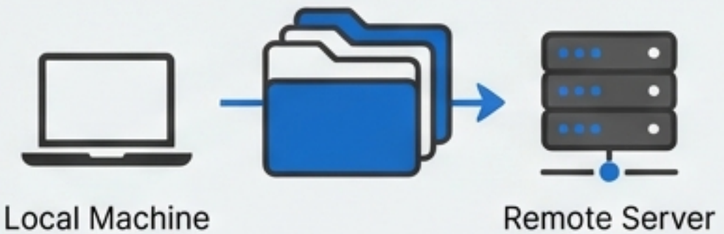
Upload
Local to Remote

```
scp results.csv user@remote.host:~/experiments/
```



Download
Remote to Local

```
scp user@remote.host:~/experiments/results.csv .
```



Directory
Recursive Directory

```
scp -r data/ user@remote.host:~/datasets/
```

Interactive Transfer (SFTP)

SFTP provides a text-based file explorer interface for browsing and moving files.

```
sftp youruserid@remote.host.edu
```

Cheat Sheet

Command	Description	Command	Description
`ls`	List files	`put local.csv`	Upload file
`cd`	Change directory	`get remote.csv`	Download file
`mkdir`	Create folder	`rm file.txt`	Delete file

Git Integration

Bubhead: Use your SSH key to authenticate clone, push, and pull operations.

Get your public key:

```
cat ~/.ssh/id_ed25519.pub
```

Profile > Preferences > SSH Keys. Paste key.

Settings > SSH and GPG keys > New SSH key. Paste key.

Clone using SSH URL:

```
git clone git@gitlab.com:group/project.git
```

Optimization: The SSH Config File

Define aliases in ~/.ssh/config to simplify your workflow.

```
Host lab-cluster
    HostName remote.host.edu
    User youruserid
    IdentityFile ~/.ssh/id_ed25519

Host gitlab.com
    User git
    IdentityFile ~/.ssh/id_ed25519
```

Instead of typing:

```
ssh youruserid@remote.host.edu
```

Just type:

```
ssh lab-cluster
```

SSH Tunneling (Port Forwarding)

Accessing internal databases securely through the firewall.

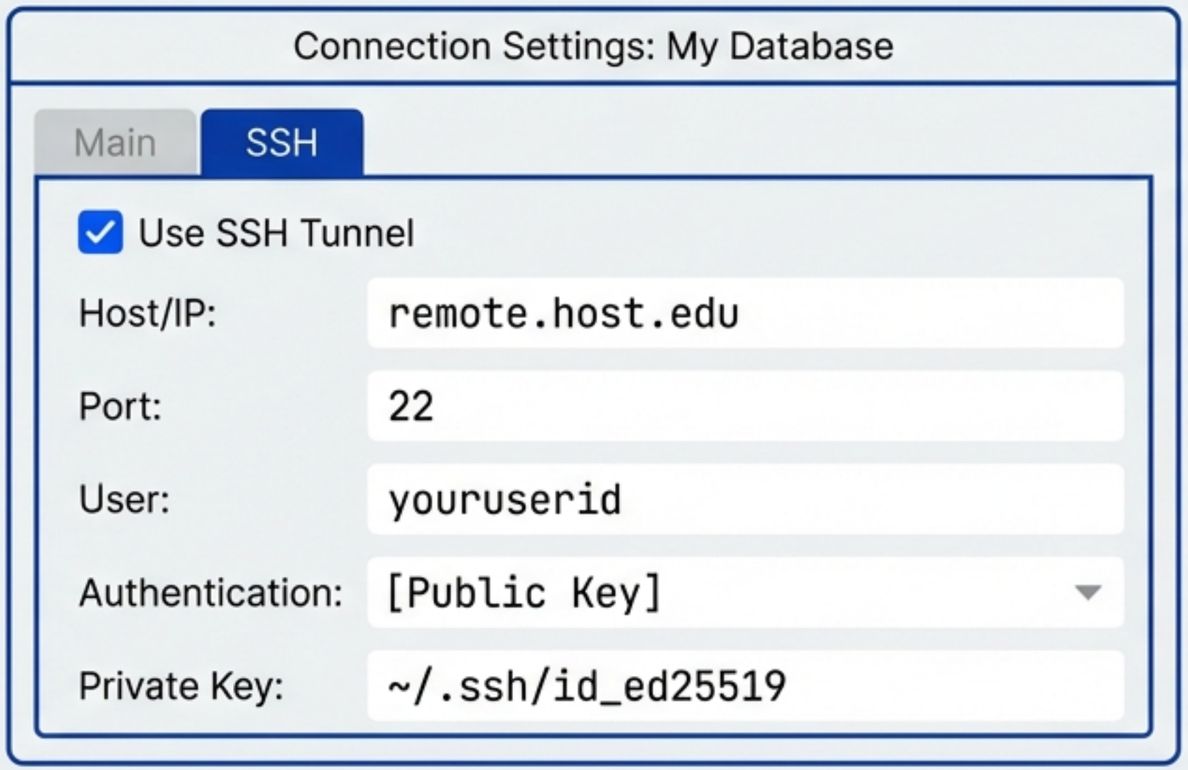

```
ssh -L 15432:localhost:5432 user@remote.host.edu
```

-L: Local Forward **15432**: Your Laptop Port

localhost:5432: Target relative to Server

Tunneling with GUI Tools (DBeaver)

Configure database clients to use SSH automatically.



DBeaver handles the tunnel automatically.

The Data Science SSH Workflow

One-Time Setup

- Generate Keys (`ssh-keygen`)
- Distribute Public Key (`ssh-copy-id`)

Daily Operations

- Connect (`ssh`)
- Transfer Data (`scp / sftp`)
- Version Control (`git pull / push`)

Advanced Access

- Tunneling (`ssh -L / DBeaver`)
- Access private resources

Best Practices & Security

Passphrases

Always set a passphrase for keys on laptops. It protects your identity if the device is stolen.

Permissions

Enforce strict permissions.

```
chmod 700 ~/.ssh
```

```
chmod 600 authorized_keys
```

Verification

Verify the host fingerprint on first connection to prevent Man-in-the-Middle attacks.

Automation starts with SSH.