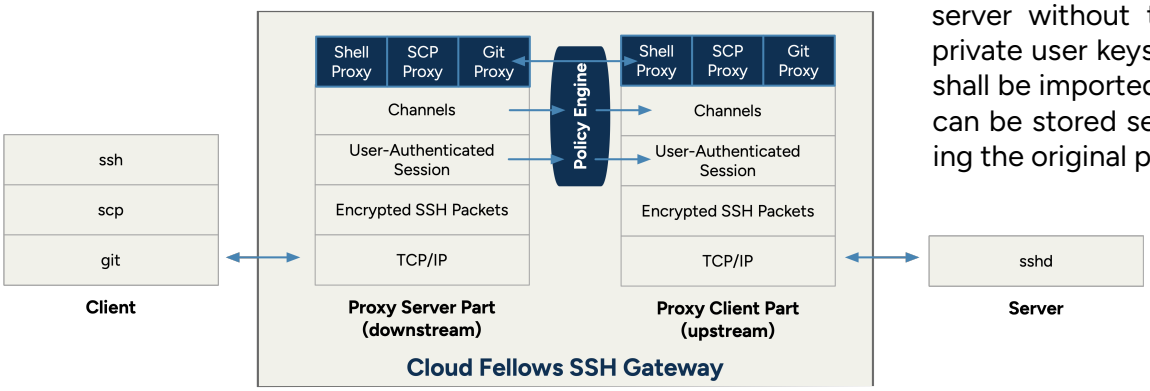
CloudFellows SSH Gateway

Deployment Modes

SSH Gateway can be deployed in Forward-Proxy and Reverse-Proxy mode, either as (virtual) appliance on-premise or as gateway in the cloud, always sits between the SSH client and the SSH server to filter the SSH traffic.



User Authentication

Support for public key authentication and user certificates from trusted CAs. Password authentication disabled by default but possible via policy.

By leveraging the SSH agent protocol, SSH Gateway can forward user authentication to the upstream server without the need to import private user keys. If private user keys shall be imported, the decryption key can be stored securely without storing the original passphrase.

Supported Application Proxies

Plain SSH protocol supported to control channel opening for commands, subsystems, TCP forward and reverse tunnels.

Shell proxy support to monitor, modify and block all commands exchanged on a shell channel.

SCP and SFTP proxies to monitor, modify or block files uploaded and downloaded; allows to filter file data chunk-by-chunk or hold chunks of data until a policy decision is made.

A git proxy to parse repository pull and push requests and filter transferred files and data.

Policy Engine

Fully flexible policy design as Python script. Define access control for which user is allowed to access with which credentials and to which destination. Optionally, leverage a built-in AuthDatabase feature to simplify the Python script.

Additional Python methods for each SSH command, shell command and transferred file via SCP, SFTP or git.

Directing user traffic to SSH Gateway

As reverse proxy, leverage DNS settings to direct traffic to SSH Gateway. As forward proxy, explicitly address SSH Gateway as a proxy with multiple options to encode the final server name in the username. Or configure SSH Gateway in virtual JumpHost mode. Simplified addressing via SSH config file.

Host Authentication

Authenticate upstream server by host keys or host certificates (from trusted CAs). Policy control for new hosts and keys. Supports SSH protocol extension for automatic host key updates.

Logging

- Access Log at the end of each SSH connection.
- Commands Log with entries for each executed SSH command.
- Files Log for each transferred file.
- Custom logs (unlimited number), triggered by policy, written at the end of a transaction.

All log files with many standard fields plus freely definable custom fields, calculated in Python policy. Log files auto-rotate daily.

Crypto Algorithms

Kex Algorithms:

- curve25519-sha256
- diffie-hellman-group14-sha256 and -sha1

Host Keys:

- ssh-ed25519
- rsa-sha2-512 and rsa-sha2-256
- ssh-ed25519-cert-v01@openssh.com
- rsa-sha2-256-cert-v01@openssh.com (and -512)

Packet Encryption:

- chacha20-poly1305@openssh.com
- aes256-ctr, aes192-ctr, aes128-ctr

Packet MAC:

- chacha20-poly1305@openssh.com
- hmac-sha2-512, hmac-sha2-256, hmac-sha1
- All also in the xxx-etm@openssh.com variant

Architecture

Single process running on Linux or MacOS
Execute either natively or within Docker container
Fixed number of working threads, asynchronous socket communication and policy task chain
Minimal system dependencies
Low system resources
Highly scalable

Cloud Fellows GmbH • Hünenweg 15 • 33104 Paderborn, Germany • sales@cloudfellows.de