SSH logins in practice: certificates vs OPKSSH

Erich Birngruber

FOSDEM’26

2026-02-01

Looking into SSH logins - why?

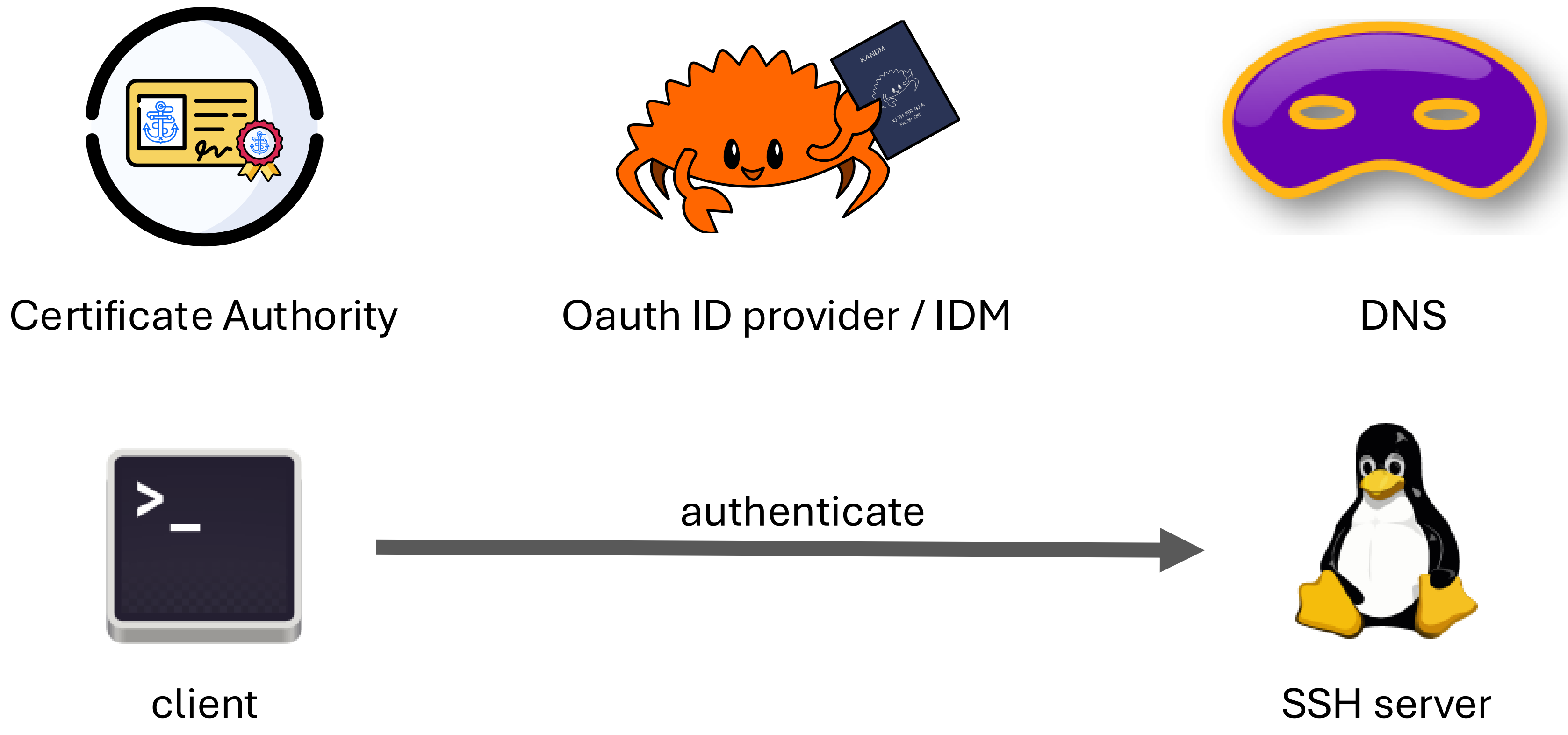
- Environment: Resarch Campus / University
- Goal: give hundrets of users access to a compute cluster
  i.e. Linux hosts
- Challenge: hosts are accessed via SSH, but many or most clients are unmanaged – unknown endpoint security

One answer to that is short lived credentials.
This can be implemented with certificates or tokens.

How are you using SSH?

The lab setup “ssh_lab”

- Container environment to with various parts of infrastructure



Demo time ...

Conclusion

Certificates

- + short lived + oauth flow
- + host validation
- + core SSH features control (forwarding, agent, etc.)

- - CA setup required

OpenPubKey SSH

- + short lived + oauth flow
- + highly customizable policies (also with claims)
- + use with multiple (existing) ID providers

- - does not cover host verification

(get over it, you need a 3rd party tool additional to SSH client)

Thank you!

Time for questions

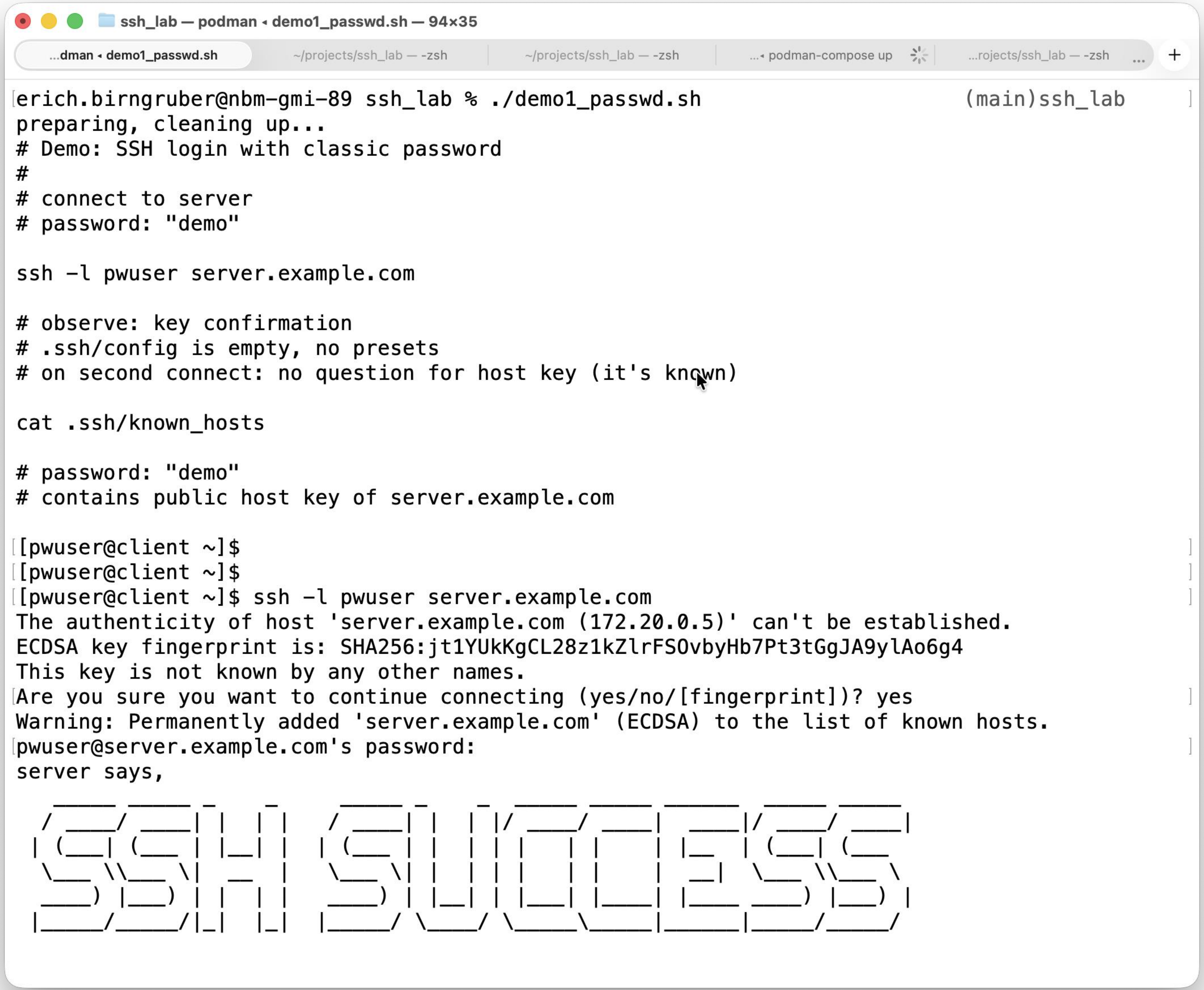
Resources

- SSH server: https://www.openssh.org
- IDM and OAuth provider: https://kanidm.github.io/kanidm/stable/
- Certificate Authority: https://smallstep.com/docs/step-ca/
- OpenPubkeySSH: https://github.com/openpubkey/opkssh
- DNS server: https://dnsmasq.org
- Container and orchestration: https://podman.io/

- The ssh_lab: https://github.com/CLIP-HPC/ssh_lab

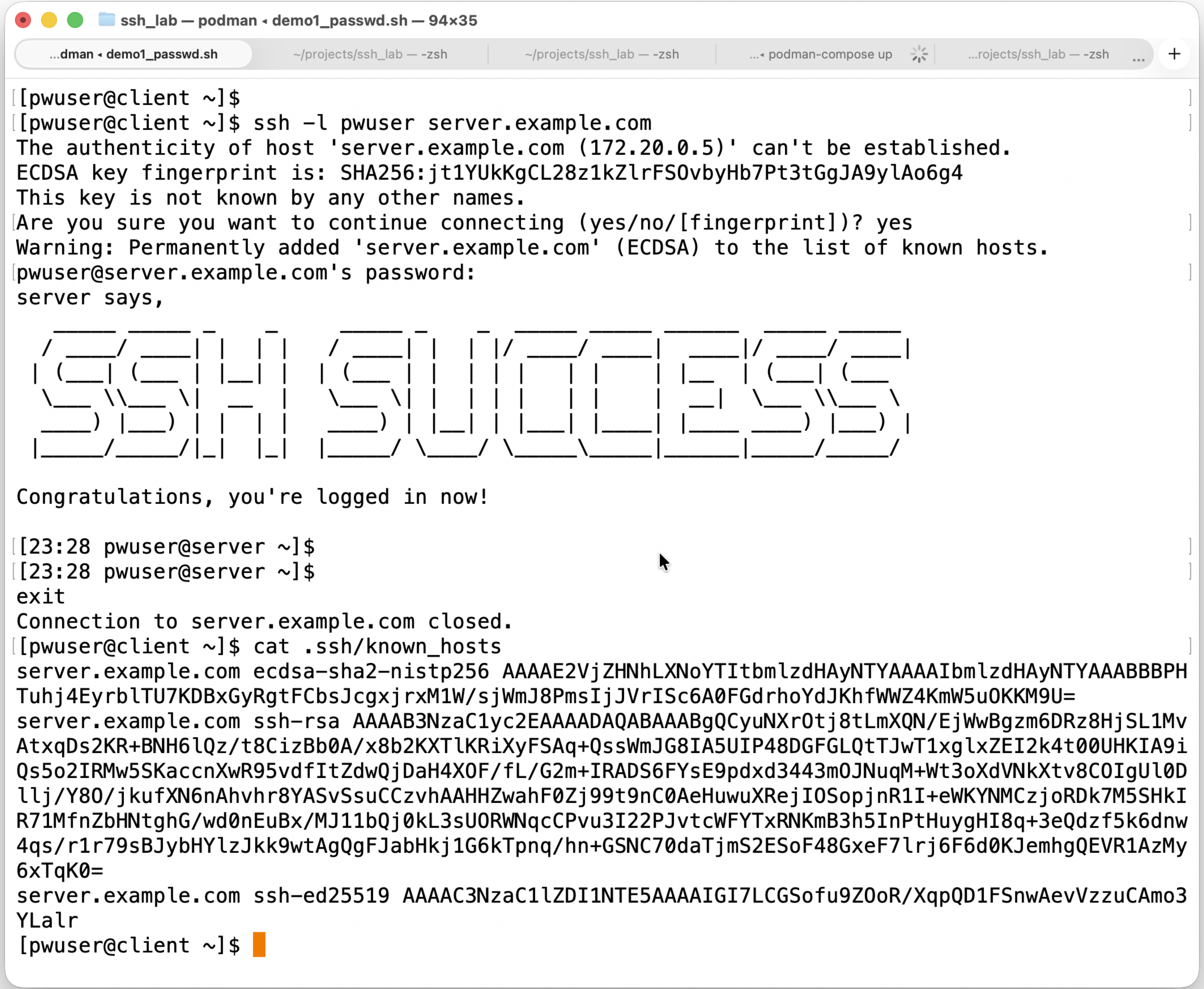
Slides of last resort

Demo 1: password



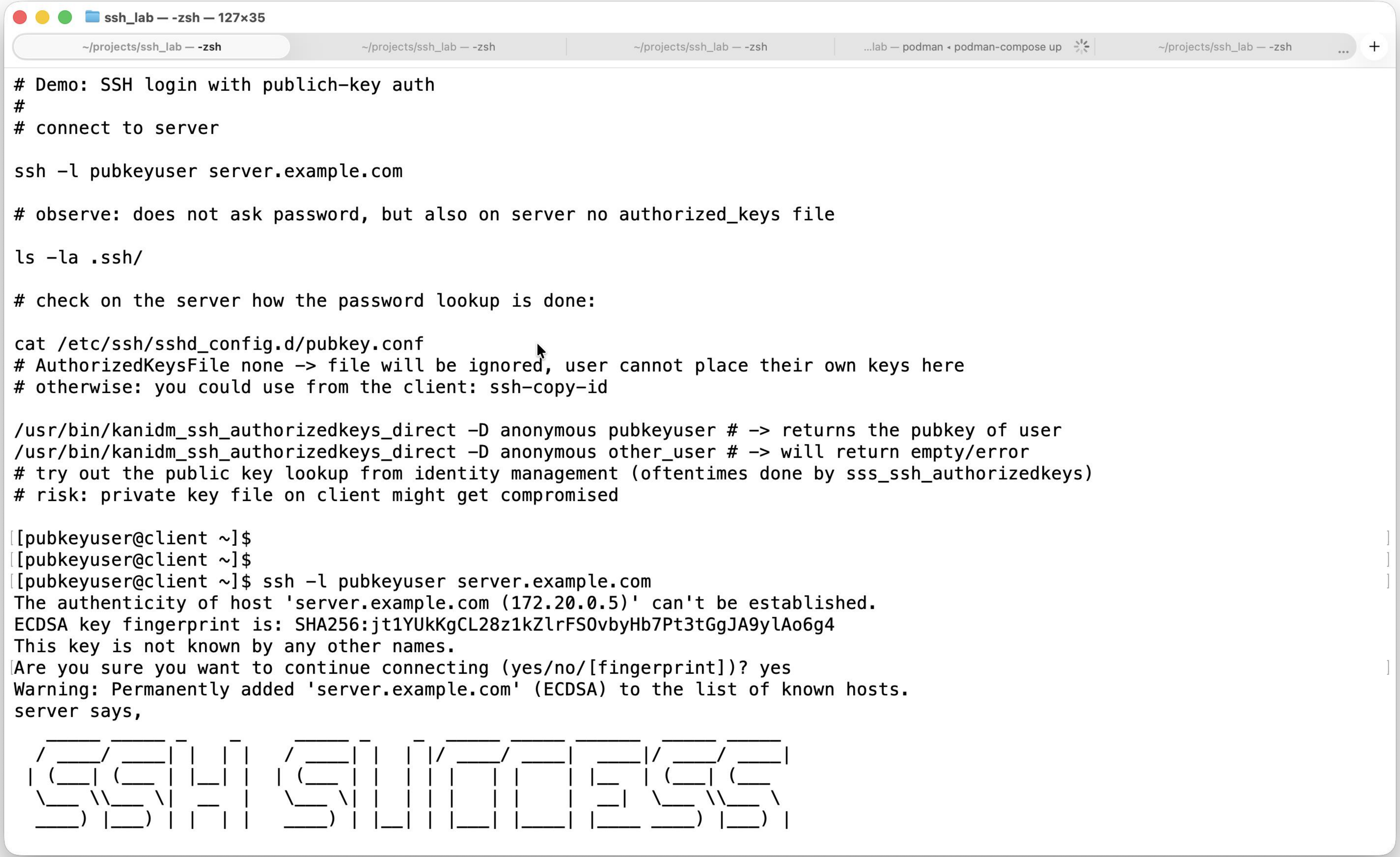
```
ssh_lab — podman ◂ demo1_passwd.sh — 94×35
```

```
erich.birngruber@nbm-gmi-89 ssh_lab % ./demo1_passwd.sh                    (main)ssh_lab
preparing, cleaning up...
# Demo: SSH login with classic password
#
# connect to server
# password: "demo"

ssh -l pwuser server.example.com

# observe: key confirmation
# .ssh/config is empty, no presets
# on second connect: no question for host key (it's known)

cat .ssh/known_hosts

# password: "demo"
# contains public host key of server.example.com

[pwuser@client ~]$
[pwuser@client ~]$
[pwuser@client ~]$ ssh -l pwuser server.example.com
The authenticity of host 'server.example.com (172.20.0.5)' can't be established.
ECDSA key fingerprint is: SHA256:jt1YUkKgCL28z1kZlrFSOvbyHb7Pt3tGgJA9ylAo6g4
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'server.example.com' (ECDSA) to the list of known hosts.
pwuser@server.example.com's password:
server says,

   _____ _____ _    _    _____ _    _  _____ _____ ______  _____ _____
  / ____/ ____| |  | |  / ____| |  | |/ ____/ ____|  ____|/ ____/ ____|
 | (___| (___ | |__| | | (___ | |  | | |   | |    | |__  | (___| (___
  \___ \\___ \|  __  |  \___ \| |  | | |   | |    |  __|  \___ \\___ \
  ____) |___) | |  | |  ____) | |__| | |___| |____| |____ ____) |___) |
 |_____/_____/|_|  |_| |_____/ \____/ \_____\_____|______|_____/_____/
```

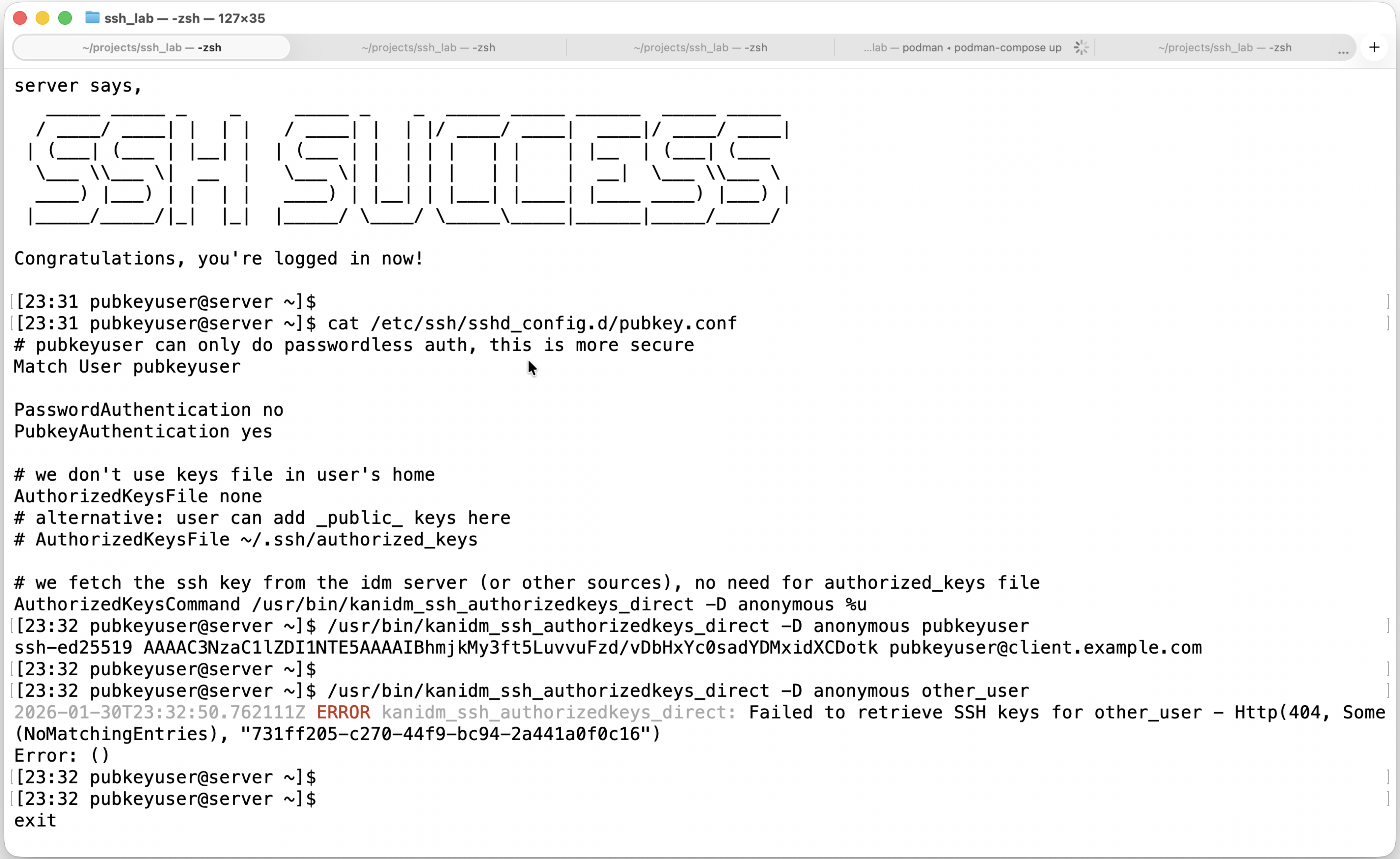

ssh_lab — podman ◂ demo1_passwd.sh — 94×35
...dman ◂ demo1_passwd.sh
~/projects/ssh_lab — -zsh
~/projects/ssh_lab — -zsh
...◂ podman-compose up
...rojects/ssh_lab — -zsh

```
[pwuser@client ~]$
[pwuser@client ~]$ ssh -l pwuser server.example.com
The authenticity of host 'server.example.com (172.20.0.5)' can't be established.
ECDSA key fingerprint is: SHA256:jt1YUkKgCL28z1kZlrFSOvbyHb7Pt3tGgJA9ylAo6g4
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'server.example.com' (ECDSA) to the list of known hosts.
pwuser@server.example.com's password:
server says,

   _____ _____ _    _    _____ _    _  _____ _____ ______  _____ _____
  / ____/ ____| |  | |  / ____| |  | |/ ____/ ____|  ____|/ ____/ ____|
 | (___| (___ | |__| | | (___ | |  | | |   | |    | |__  | (___| (___
  \___ \\___ \|  __  |  \___ \| |  | | |   | |    |  __|  \___ \\___ \
  ____) |___) | |  | |  ____) | |__| | |___| |____| |____ ____) |___) |
 |_____/_____/|_|  |_| |_____/ \____/ \_____\_____|______|_____/_____/

Congratulations, you're logged in now!

[23:28 pwuser@server ~]$
[23:28 pwuser@server ~]$
exit
Connection to server.example.com closed.
[pwuser@client ~]$ cat .ssh/known_hosts
server.example.com ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBPH
Tuhj4EyrblTU7KDBxGyRgtFCbsJcgxjrxM1W/sjWmJ8PmsIjJVrISc6A0FGdrhoYdJKhfWWZ4KmW5uOKKM9U=
server.example.com ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQCyuNXrOtj8tLmXQN/EjWwBgzm6DRz8HjSL1Mv
AtxqDs2KR+BNH6lQz/t8CizBb0A/x8b2KXTlKRiXyFSAq+QssWmJG8IA5UIP48DGFGLQtTJwT1xglxZEI2k4t00UHKIA9i
Qs5o2IRMw5SKaccnXwR95vdfItZdwQjDaH4XOF/fL/G2m+IRADS6FYsE9pdxd3443mOJNuqM+Wt3oXdVNkXtv8COIgUl0D
llj/Y8O/jkufXN6nAhvhr8YASvSsuCCzvhAAHHZwahF0Zj99t9nC0AeHuwuXRejIOSopjnR1I+eWKYNMCzjoRDk7M5SHkI
R71MfnZbHNtghG/wd0nEuBx/MJ11bQj0kL3sUORWNqcCPvu3I22PJvtcWFYTxRNKmB3h5InPtHuygHI8q+3eQdzf5k6dnw
4qs/r1r79sBJybHYlzJkk9wtAgQgFJabHkj1G6kTpnq/hn+GSNC70daTjmS2ESoF48GxeF7lrj6F6d0KJemhgQEVR1AzMy
6xTqK0=
server.example.com ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIGI7LCGSofu9ZOoR/XqpQD1FSnwAevVzzuCAmo3
YLalr
[pwuser@client ~]$
```

Demo 2: pubkey

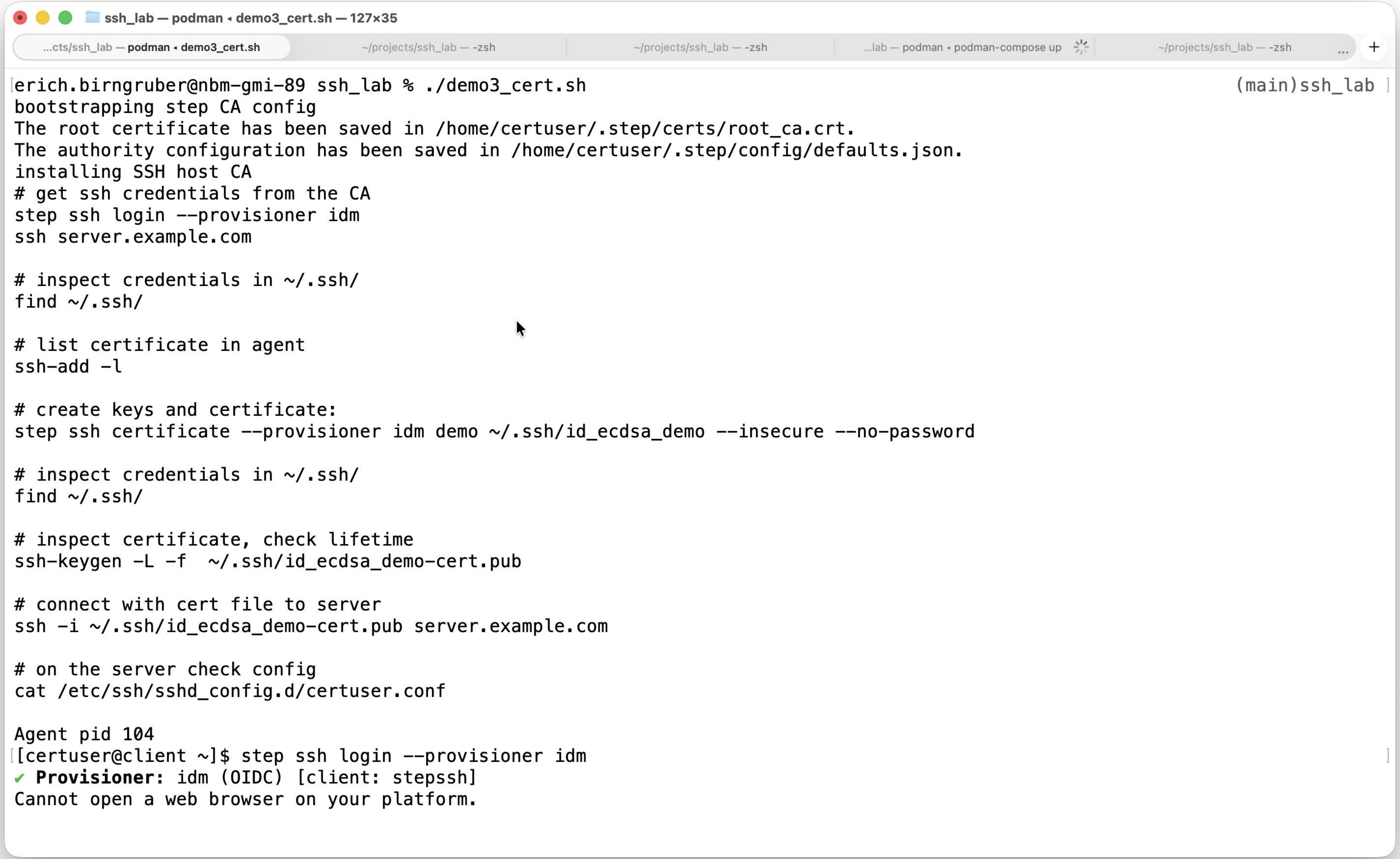

ssh_lab — -zsh — 127×35
~/projects/ssh_lab — -zsh
~/projects/ssh_lab — -zsh
~/projects/ssh_lab — -zsh
...lab — podman ◂ podman-compose up
~/projects/ssh_lab — -zsh

```
# Demo: SSH login with publich-key auth
#
# connect to server

ssh -l pubkeyuser server.example.com

# observe: does not ask password, but also on server no authorized_keys file

ls -la .ssh/

# check on the server how the password lookup is done:

cat /etc/ssh/sshd_config.d/pubkey.conf
# AuthorizedKeysFile none -> file will be ignored, user cannot place their own keys here
# otherwise: you could use from the client: ssh-copy-id

/usr/bin/kanidm_ssh_authorizedkeys_direct -D anonymous pubkeyuser # -> returns the pubkey of user
/usr/bin/kanidm_ssh_authorizedkeys_direct -D anonymous other_user # -> will return empty/error
# try out the public key lookup from identity management (oftentimes done by sss_ssh_authorizedkeys)
# risk: private key file on client might get compromised

[pubkeyuser@client ~]$
[pubkeyuser@client ~]$
[pubkeyuser@client ~]$ ssh -l pubkeyuser server.example.com
The authenticity of host 'server.example.com (172.20.0.5)' can't be established.
ECDSA key fingerprint is: SHA256:jt1YUkKgCL28z1kZlrFSOvbyHb7Pt3tGgJA9ylAo6g4
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'server.example.com' (ECDSA) to the list of known hosts.
server says,

   _____ _____ _    _    _____ _    _  _____ _____ ______  _____ _____
  / ____/ ____| |  | |  / ____| |  | |/ ____/ ____|  ____|/ ____/ ____|
 | (___| (___ | |__| | | (___ | |  | | |   | |    | |__  | (___| (___
  \___ \\___ \|  __  |  \___ \| |  | | |   | |    |  __|  \___ \\___ \
  ____) |___) | |  | |  ____) | |__| | |___| |____| |____ ____) |___) |
```

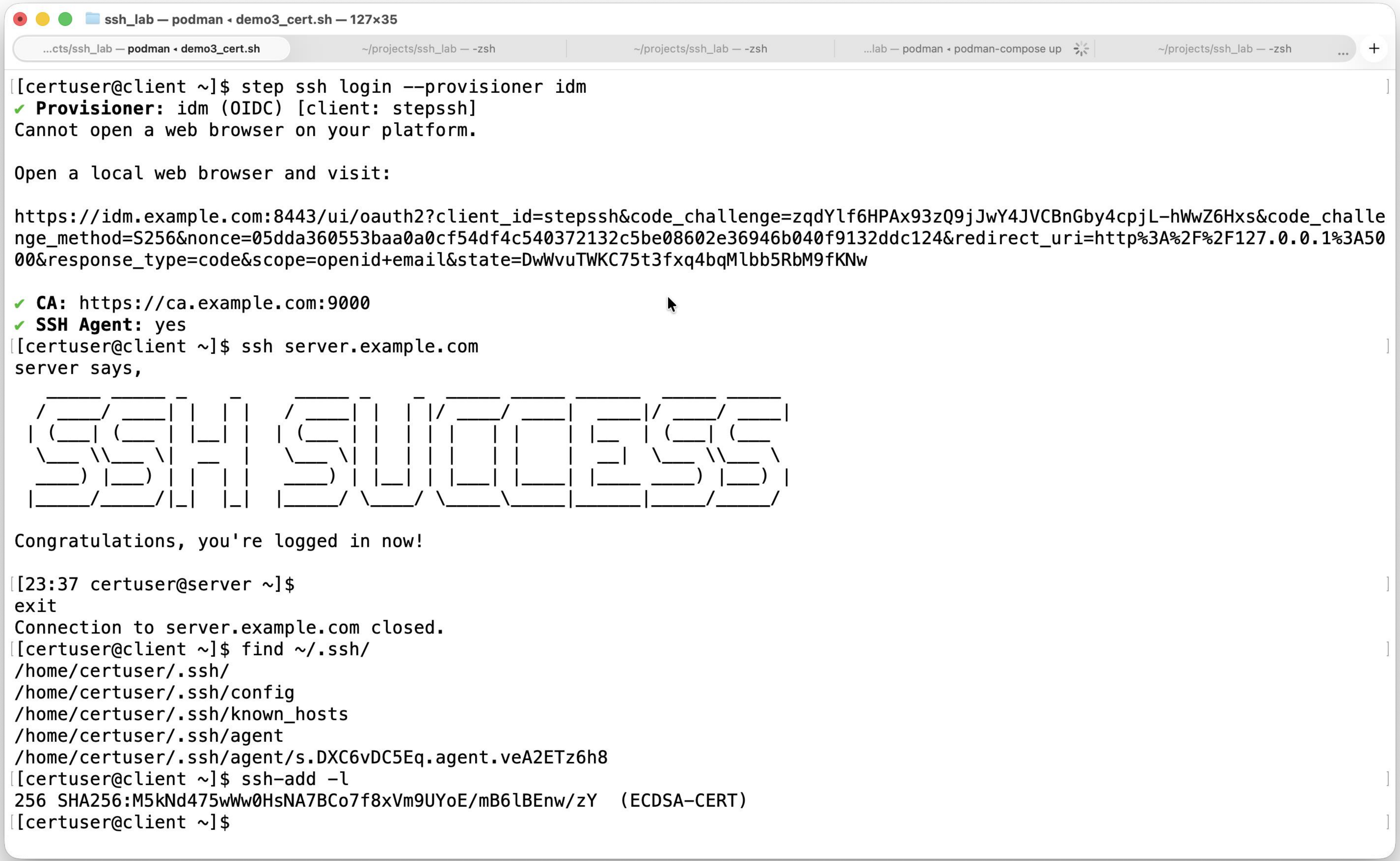

ssh_lab — -zsh — 127×35
~/projects/ssh_lab — -zsh
~/projects/ssh_lab — -zsh
~/projects/ssh_lab — -zsh
...lab — podman ◂ podman-compose up
~/projects/ssh_lab — -zsh

```
server says,
   _____ _____ _    _    _____ _    _  _____ _____ ______  _____ _____ 
  / ____/ ____| |  | |  / ____| |  | |/ ____/ ____|  ____|/ ____/ ____|
 | (___| (___ | |__| | | (___ | |  | | |   | |    | |__  | (___| (___  
  \___ \\___ \|  __  |  \___ \| |  | | |   | |    |  __|  \___ \\___ \ 
  ____) |___) | |  | |  ____) | |__| | |___| |____| |____ ____) |___) |
 |_____/_____/|_|  |_| |_____/ \____/ \_____\_____|______|_____/_____/ 

Congratulations, you're logged in now!

[23:31 pubkeyuser@server ~]$
[23:31 pubkeyuser@server ~]$ cat /etc/ssh/sshd_config.d/pubkey.conf
# pubkeyuser can only do passwordless auth, this is more secure
Match User pubkeyuser

PasswordAuthentication no
PubkeyAuthentication yes

# we don't use keys file in user's home
AuthorizedKeysFile none
# alternative: user can add _public_ keys here
# AuthorizedKeysFile ~/.ssh/authorized_keys

# we fetch the ssh key from the idm server (or other sources), no need for authorized_keys file
AuthorizedKeysCommand /usr/bin/kanidm_ssh_authorizedkeys_direct -D anonymous %u
[23:32 pubkeyuser@server ~]$ /usr/bin/kanidm_ssh_authorizedkeys_direct -D anonymous pubkeyuser
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIBhmjkMy3ft5LuvvuFzd/vDbHxYc0sadYDMxidXCDotk pubkeyuser@client.example.com
[23:32 pubkeyuser@server ~]$
[23:32 pubkeyuser@server ~]$ /usr/bin/kanidm_ssh_authorizedkeys_direct -D anonymous other_user
2026-01-30T23:32:50.762111Z ERROR kanidm_ssh_authorizedkeys_direct: Failed to retrieve SSH keys for other_user - Http(404, Some
(NoMatchingEntries), "731ff205-c270-44f9-bc94-2a441a0f0c16")
Error: ()
[23:32 pubkeyuser@server ~]$
[23:32 pubkeyuser@server ~]$
exit
```

Demo 3: certificates

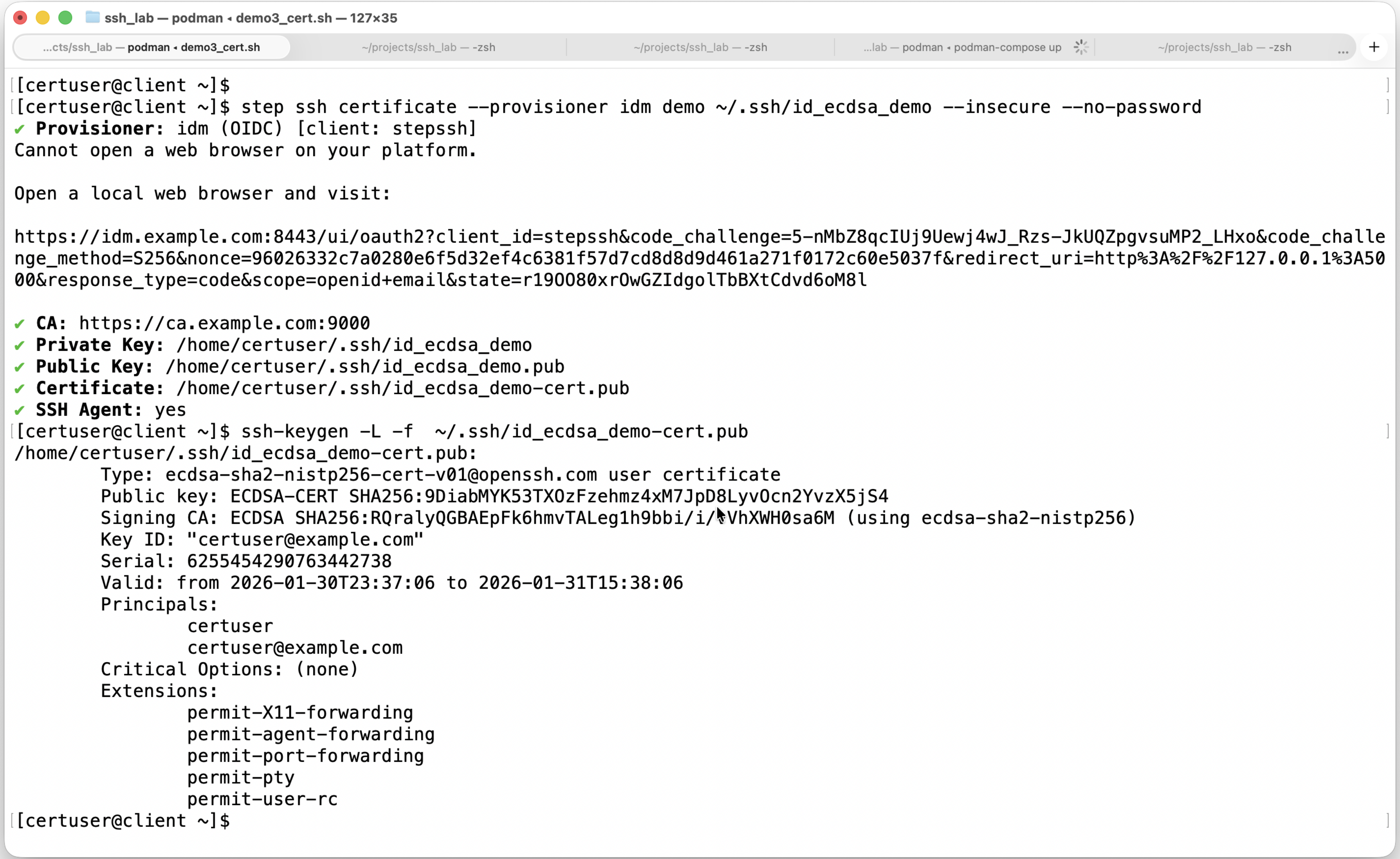

ssh_lab — podman ◂ demo3_cert.sh — 127×35
...cts/ssh_lab — podman ◂ demo3_cert.sh
~/projects/ssh_lab — -zsh
~/projects/ssh_lab — -zsh
...lab — podman ◂ podman-compose up
~/projects/ssh_lab — -zsh

```
erich.birngruber@nbm-gmi-89 ssh_lab % ./demo3_cert.sh                                                                  (main)ssh_lab
bootstrapping step CA config
The root certificate has been saved in /home/certuser/.step/certs/root_ca.crt.
The authority configuration has been saved in /home/certuser/.step/config/defaults.json.
installing SSH host CA
# get ssh credentials from the CA
step ssh login --provisioner idm
ssh server.example.com

# inspect credentials in ~/.ssh/
find ~/.ssh/

# list certificate in agent
ssh-add -l

# create keys and certificate:
step ssh certificate --provisioner idm demo ~/.ssh/id_ecdsa_demo --insecure --no-password

# inspect credentials in ~/.ssh/
find ~/.ssh/

# inspect certificate, check lifetime
ssh-keygen -L -f  ~/.ssh/id_ecdsa_demo-cert.pub

# connect with cert file to server
ssh -i ~/.ssh/id_ecdsa_demo-cert.pub server.example.com

# on the server check config
cat /etc/ssh/sshd_config.d/certuser.conf

Agent pid 104
[certuser@client ~]$ step ssh login --provisioner idm
✔ Provisioner: idm (OIDC) [client: stepssh]
Cannot open a web browser on your platform.
```

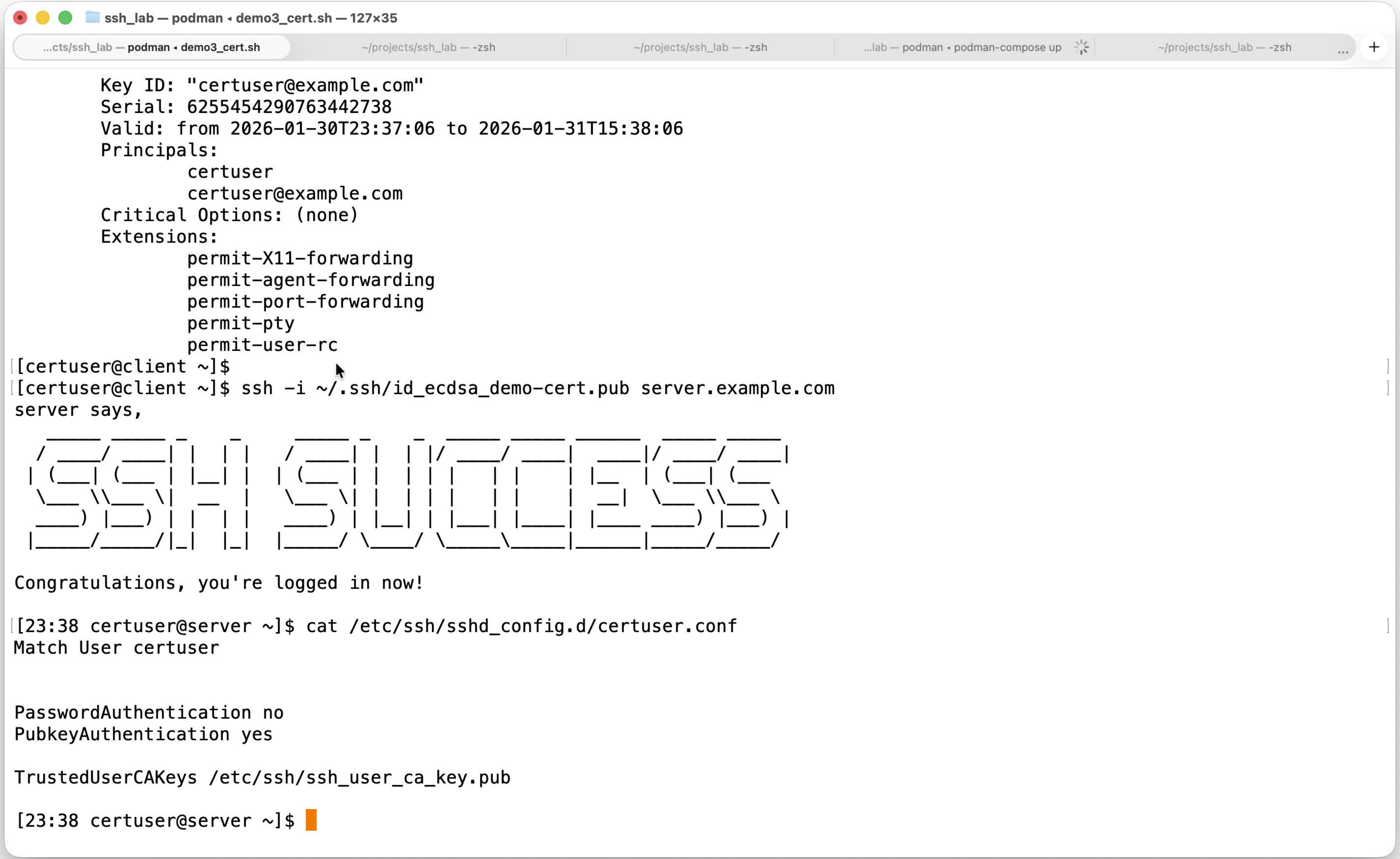


ssh_lab — podman ‹ demo3_cert.sh — 127×35
...cts/ssh_lab — podman ‹ demo3_cert.sh
~/projects/ssh_lab — -zsh
~/projects/ssh_lab — -zsh
...lab — podman ‹ podman-compose up
~/projects/ssh_lab — -zsh

```
[certuser@client ~]$ step ssh login --provisioner idm
✔ Provisioner: idm (OIDC) [client: stepssh]
Cannot open a web browser on your platform.

Open a local web browser and visit:

https://idm.example.com:8443/ui/oauth2?client_id=stepssh&code_challenge=zqdYlf6HPAx93zQ9jJwY4JVCBnGby4cpjL-hWwZ6Hxs&code_challe
nge_method=S256&nonce=05dda360553baa0a0cf54df4c540372132c5be08602e36946b040f9132ddc124&redirect_uri=http%3A%2F%2F127.0.0.1%3A50
00&response_type=code&scope=openid+email&state=DwWvuTWKC75t3fxq4bqMlbb5RbM9fKNw

✔ CA: https://ca.example.com:9000
✔ SSH Agent: yes
[certuser@client ~]$ ssh server.example.com
server says,
   _____ _____ _    _    _____ _    _  _____ _____ ______  _____ _____
  / ____/ ____| |  | |  / ____| |  | |/ ____/ ____|  ____|/ ____/ ____|
 | (___| (___ | |__| | | (___ | |  | | |   | |    | |__  | (___| (___
  \___ \\___ \|  __  |  \___ \| |  | | |   | |    |  __|  \___ \\___ \
  ____) |___) | |  | |  ____) | |__| | |___| |____| |____ ____) |___) |
 |_____/_____/|_|  |_| |_____/ \____/ \_____\_____|______|_____/_____/

Congratulations, you're logged in now!

[23:37 certuser@server ~]$
exit
Connection to server.example.com closed.
[certuser@client ~]$ find ~/.ssh/
/home/certuser/.ssh/
/home/certuser/.ssh/config
/home/certuser/.ssh/known_hosts
/home/certuser/.ssh/agent
/home/certuser/.ssh/agent/s.DXC6vDC5Eq.agent.veA2ETz6h8
[certuser@client ~]$ ssh-add -l
256 SHA256:M5kNd475wWw0HsNA7BCo7f8xVm9UYoE/mB6lBEnw/zY  (ECDSA-CERT)
[certuser@client ~]$
```

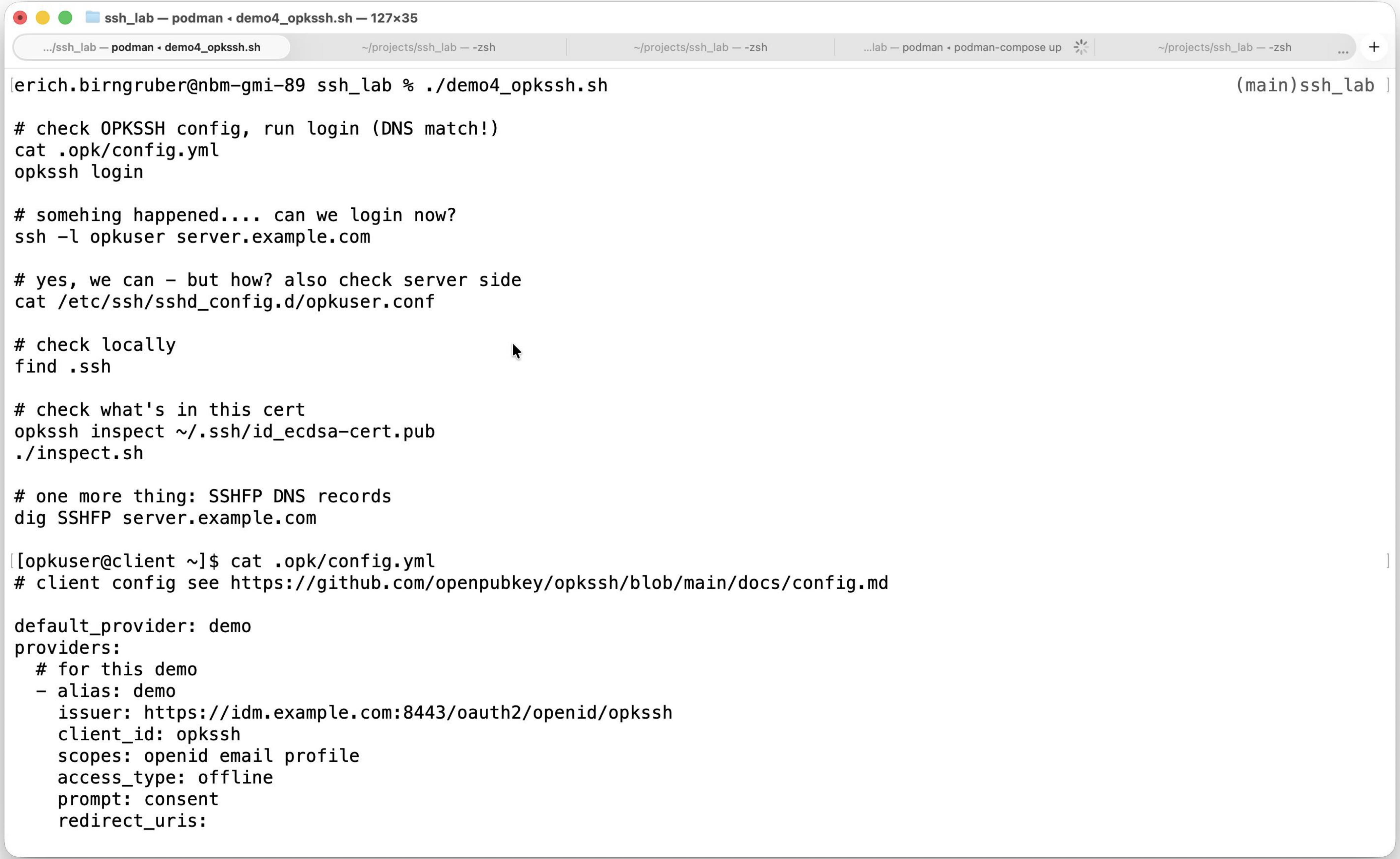

ssh_lab — podman ◂ demo3_cert.sh — 127×35
...cts/ssh_lab — podman ◂ demo3_cert.sh
~/projects/ssh_lab — -zsh
~/projects/ssh_lab — -zsh
...lab — podman ◂ podman-compose up
~/projects/ssh_lab — -zsh

```
[certuser@client ~]$
[certuser@client ~]$ step ssh certificate --provisioner idm demo ~/.ssh/id_ecdsa_demo --insecure --no-password
✔ Provisioner: idm (OIDC) [client: stepssh]
Cannot open a web browser on your platform.

Open a local web browser and visit:

https://idm.example.com:8443/ui/oauth2?client_id=stepssh&code_challenge=5-nMbZ8qcIUj9Uewj4wJ_Rzs-JkUQZpgvsuMP2_LHxo&code_challe
nge_method=S256&nonce=96026332c7a0280e6f5d32ef4c6381f57d7cd8d8d9d461a271f0172c60e5037f&redirect_uri=http%3A%2F%2F127.0.0.1%3A50
00&response_type=code&scope=openid+email&state=r19O080xrOwGZIdgolTbBXtCdvd6oM8l

✔ CA: https://ca.example.com:9000
✔ Private Key: /home/certuser/.ssh/id_ecdsa_demo
✔ Public Key: /home/certuser/.ssh/id_ecdsa_demo.pub
✔ Certificate: /home/certuser/.ssh/id_ecdsa_demo-cert.pub
✔ SSH Agent: yes
[certuser@client ~]$ ssh-keygen -L -f  ~/.ssh/id_ecdsa_demo-cert.pub
/home/certuser/.ssh/id_ecdsa_demo-cert.pub:
        Type: ecdsa-sha2-nistp256-cert-v01@openssh.com user certificate
        Public key: ECDSA-CERT SHA256:9DiabMYK53TXOzFzehmz4xM7JpD8LyvOcn2YvzX5jS4
        Signing CA: ECDSA SHA256:RQralyQGBAEpFk6hmvTALeg1h9bbi/i/tVhXWH0sa6M (using ecdsa-sha2-nistp256)
        Key ID: "certuser@example.com"
        Serial: 6255454290763442738
        Valid: from 2026-01-30T23:37:06 to 2026-01-31T15:38:06
        Principals:
                certuser
                certuser@example.com
        Critical Options: (none)
        Extensions:
                permit-X11-forwarding
                permit-agent-forwarding
                permit-port-forwarding
                permit-pty
                permit-user-rc
[certuser@client ~]$
```

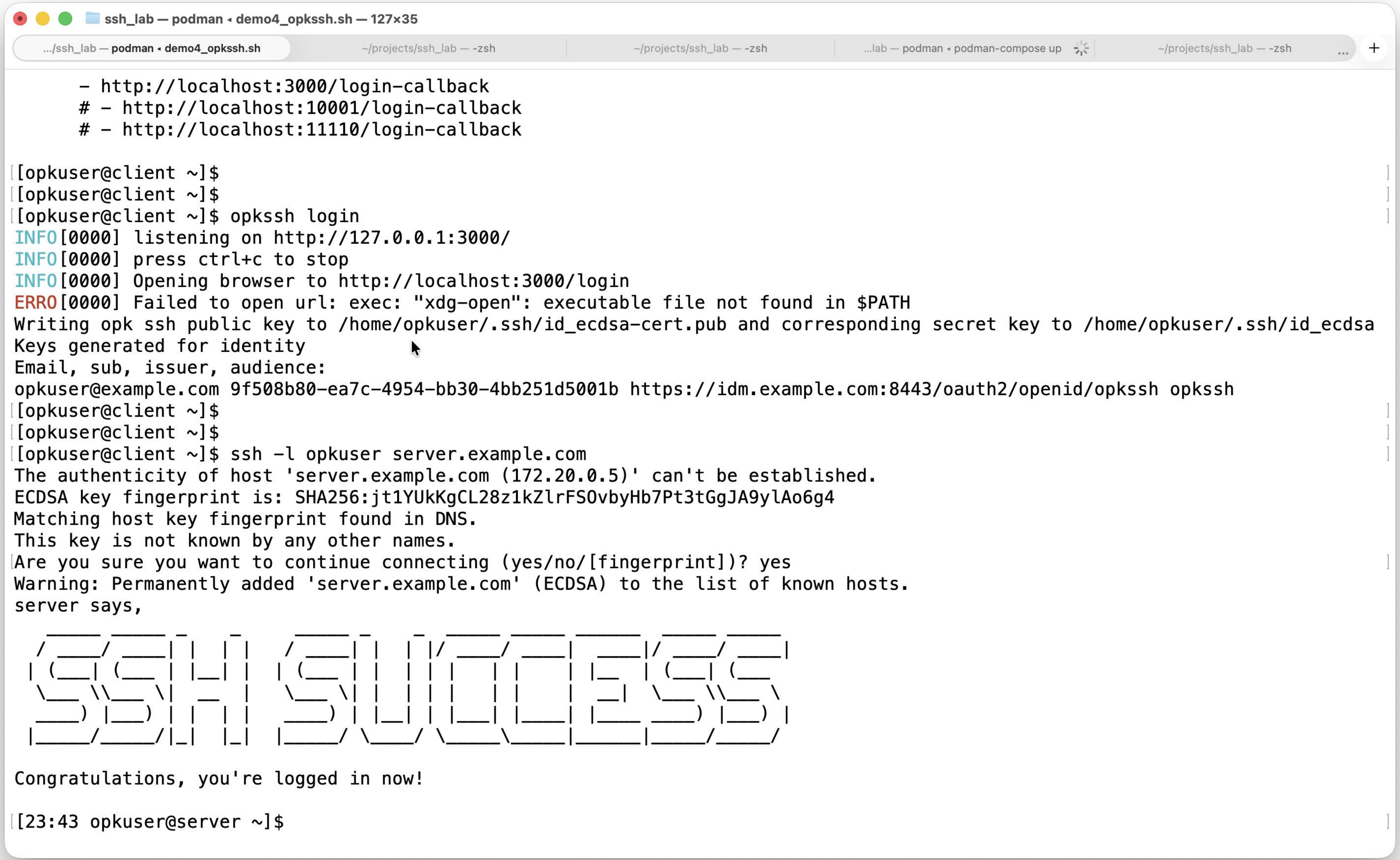

ssh_lab — podman ◂ demo3_cert.sh — 127×35
...cts/ssh_lab — podman ◂ demo3_cert.sh
~/projects/ssh_lab — -zsh
~/projects/ssh_lab — -zsh
...lab — podman ◂ podman-compose up
~/projects/ssh_lab — -zsh

```
        Key ID: "certuser@example.com"
        Serial: 6255454290763442738
        Valid: from 2026-01-30T23:37:06 to 2026-01-31T15:38:06
        Principals:
                certuser
                certuser@example.com
        Critical Options: (none)
        Extensions:
                permit-X11-forwarding
                permit-agent-forwarding
                permit-port-forwarding
                permit-pty
                permit-user-rc
[certuser@client ~]$
[certuser@client ~]$ ssh -i ~/.ssh/id_ecdsa_demo-cert.pub server.example.com
server says,

   _____ _____ _    _    _____ _    _  _____ _____ ______  _____ _____
  / ____/ ____| |  | |  / ____| |  | |/ ____/ ____|  ____|/ ____/ ____|
 | (___| (___ | |__| | | (___ | |  | | |   | |    | |__  | (___| (___
  \___ \\___ \|  __  |  \___ \| |  | | |   | |    |  __|  \___ \\___ \
  ____) |___) | |  | |  ____) | |__| | |___| |____| |____ ____) |___) |
 |_____/_____/|_|  |_| |_____/ \____/ \_____\_____|______|_____/_____/

Congratulations, you're logged in now!

[23:38 certuser@server ~]$ cat /etc/ssh/sshd_config.d/certuser.conf
Match User certuser


PasswordAuthentication no
PubkeyAuthentication yes

TrustedUserCAKeys /etc/ssh/ssh_user_ca_key.pub

[23:38 certuser@server ~]$
```

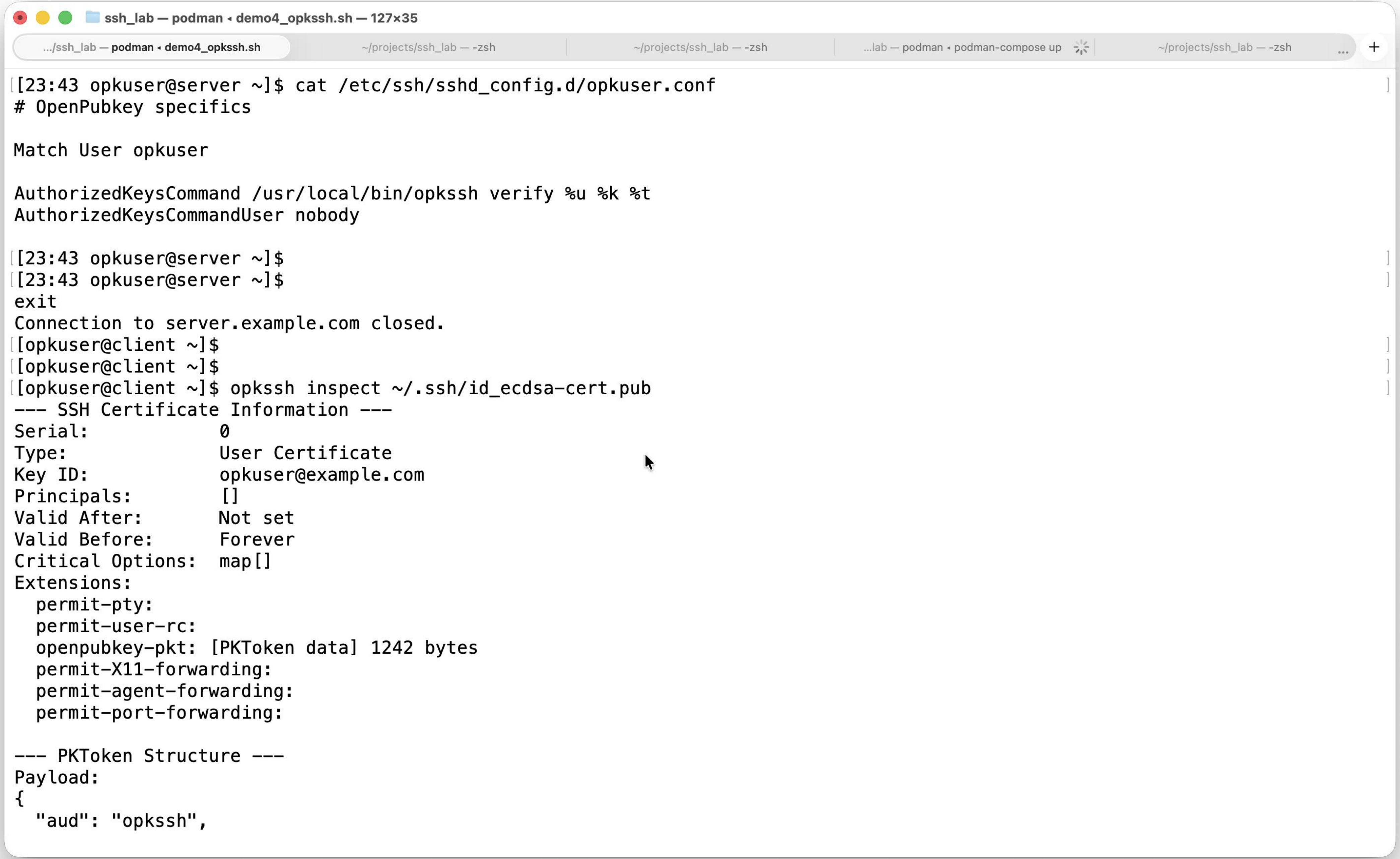
Demo 4: OPKSSH


ssh_lab — podman ◂ demo4_opkssh.sh — 127×35
.../ssh_lab — podman ◂ demo4_opkssh.sh
~/projects/ssh_lab — -zsh
~/projects/ssh_lab — -zsh
...lab — podman ◂ podman-compose up
~/projects/ssh_lab — -zsh

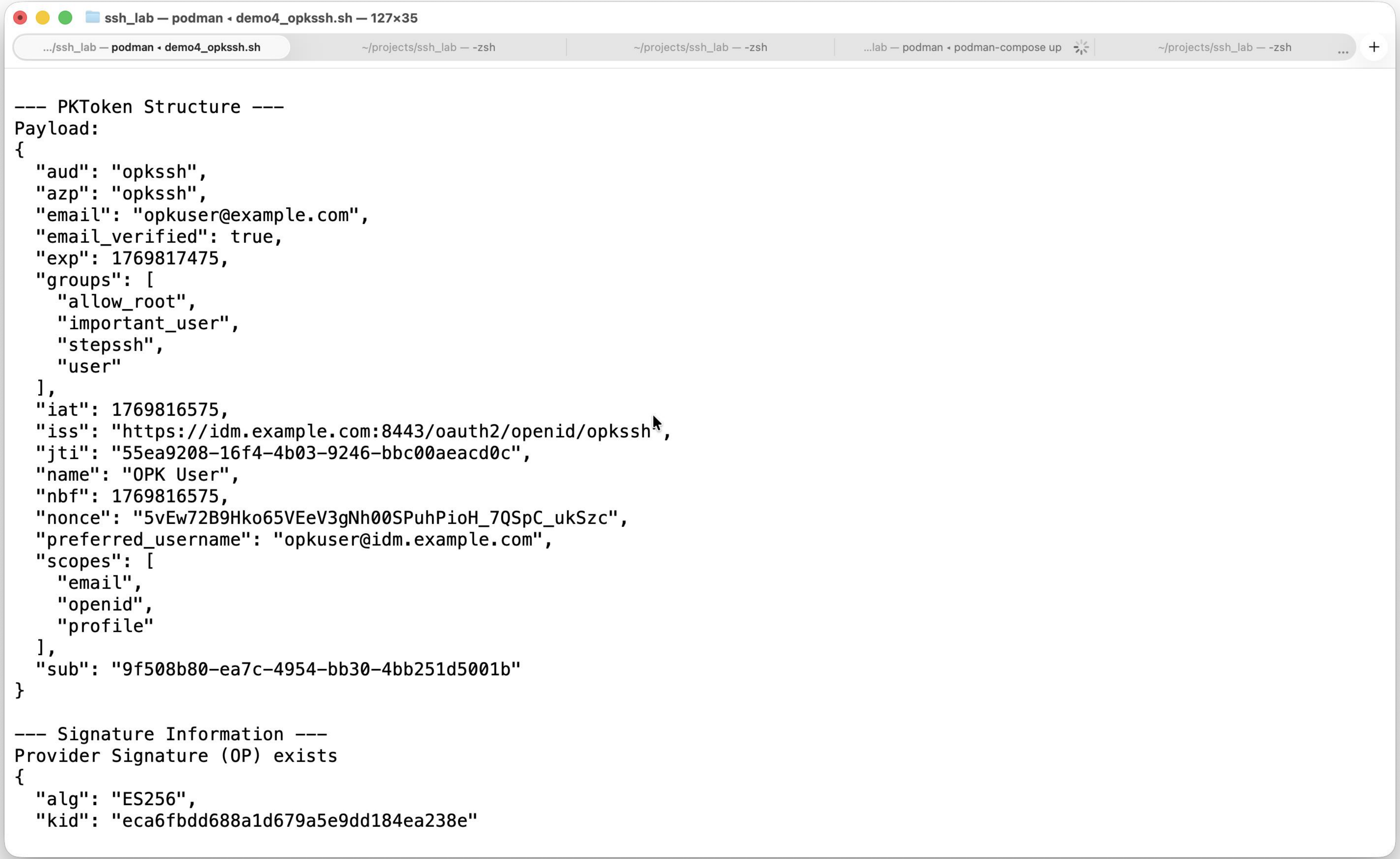
```
erich.birngruber@nbm-gmi-89 ssh_lab % ./demo4_opkssh.sh                                     (main)ssh_lab

# check OPKSSH config, run login (DNS match!)
cat .opk/config.yml
opkssh login

# somehing happened.... can we login now?
ssh -l opkuser server.example.com

# yes, we can - but how? also check server side
cat /etc/ssh/sshd_config.d/opkuser.conf

# check locally
find .ssh

# check what's in this cert
opkssh inspect ~/.ssh/id_ecdsa-cert.pub
./inspect.sh

# one more thing: SSHFP DNS records
dig SSHFP server.example.com

[opkuser@client ~]$ cat .opk/config.yml
# client config see https://github.com/openpubkey/opkssh/blob/main/docs/config.md

default_provider: demo
providers:
  # for this demo
  - alias: demo
    issuer: https://idm.example.com:8443/oauth2/openid/opkssh
    client_id: opkssh
    scopes: openid email profile
    access_type: offline
    prompt: consent
    redirect_uris:
```


ssh_lab — podman ◂ demo4_opkssh.sh — 127×35
.../ssh_lab — podman ◂ demo4_opkssh.sh
~/projects/ssh_lab — -zsh
~/projects/ssh_lab — -zsh
...lab — podman ◂ podman-compose up
~/projects/ssh_lab — -zsh

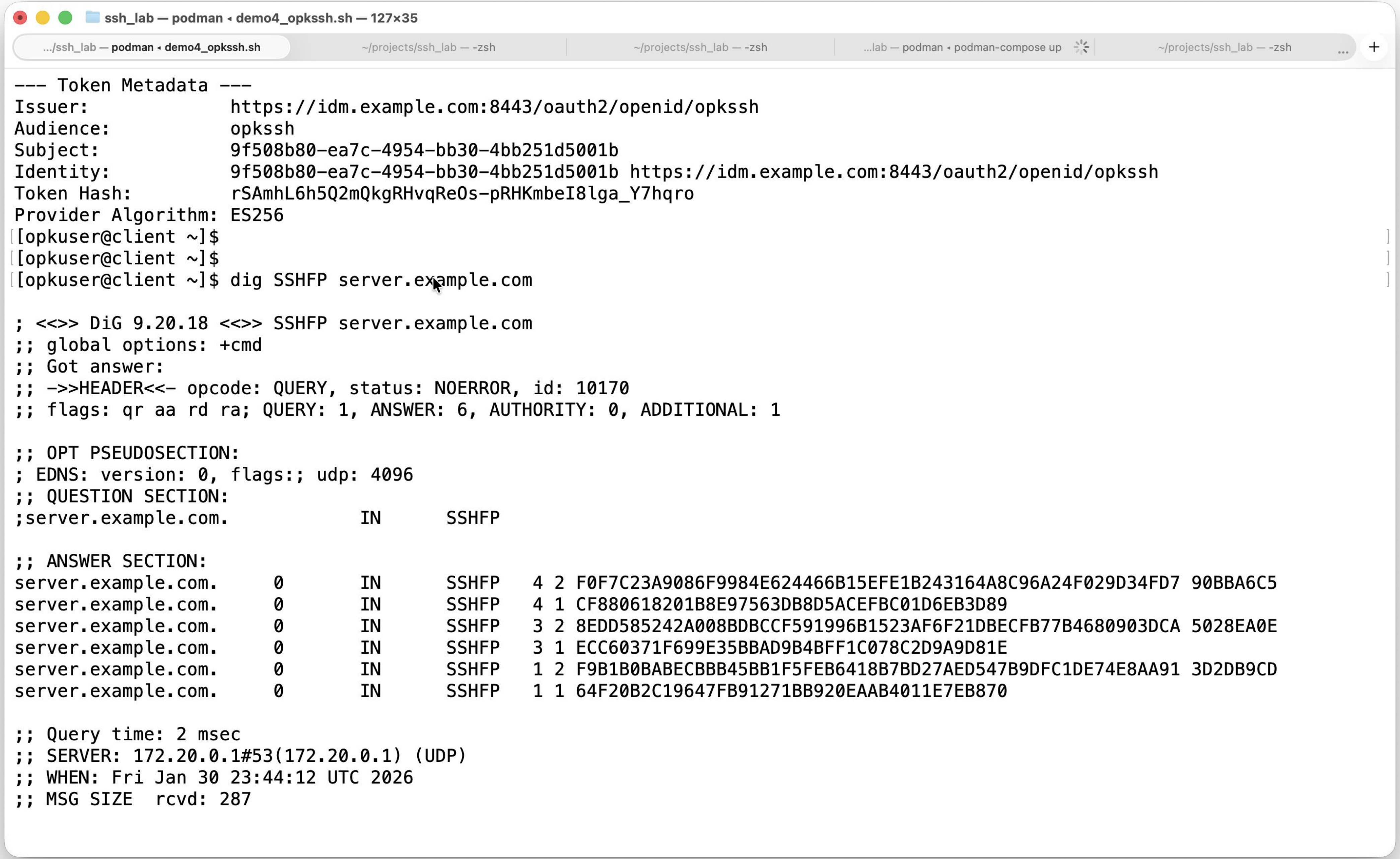
```
      - http://localhost:3000/login-callback
      # - http://localhost:10001/login-callback
      # - http://localhost:11110/login-callback

[opkuser@client ~]$
[opkuser@client ~]$
[opkuser@client ~]$ opkssh login
INFO[0000] listening on http://127.0.0.1:3000/
INFO[0000] press ctrl+c to stop
INFO[0000] Opening browser to http://localhost:3000/login
ERRO[0000] Failed to open url: exec: "xdg-open": executable file not found in $PATH
Writing opk ssh public key to /home/opkuser/.ssh/id_ecdsa-cert.pub and corresponding secret key to /home/opkuser/.ssh/id_ecdsa
Keys generated for identity
Email, sub, issuer, audience:
opkuser@example.com 9f508b80-ea7c-4954-bb30-4bb251d5001b https://idm.example.com:8443/oauth2/openid/opkssh opkssh
[opkuser@client ~]$
[opkuser@client ~]$
[opkuser@client ~]$ ssh -l opkuser server.example.com
The authenticity of host 'server.example.com (172.20.0.5)' can't be established.
ECDSA key fingerprint is: SHA256:jt1YUkKgCL28z1kZlrFSOvbyHb7Pt3tGgJA9ylAo6g4
Matching host key fingerprint found in DNS.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'server.example.com' (ECDSA) to the list of known hosts.
server says,

   _____ _____ _    _    _____ _    _  _____ _____ ______  _____ _____
  / ____/ ____| |  | |  / ____| |  | |/ ____/ ____|  ____|/ ____/ ____|
 | (___| (___ | |__| | | (___ | |  | | |   | |    | |__  | (___| (___
  \___ \\___ \|  __  |  \___ \| |  | | |   | |    |  __|  \___ \\___ \
  ____) |___) | |  | |  ____) | |__| | |___| |____| |____ ____) |___) |
 |_____/_____/|_|  |_| |_____/ \____/ \_____\_____|______|_____/_____/

Congratulations, you're logged in now!

[23:43 opkuser@server ~]$
```


ssh_lab — podman ‹ demo4_opkssh.sh — 127×35
.../ssh_lab — podman ‹ demo4_opkssh.sh
~/projects/ssh_lab — -zsh
~/projects/ssh_lab — -zsh
...lab — podman ‹ podman-compose up
~/projects/ssh_lab — -zsh

```
[23:43 opkuser@server ~]$ cat /etc/ssh/sshd_config.d/opkuser.conf
# OpenPubkey specifics

Match User opkuser

AuthorizedKeysCommand /usr/local/bin/opkssh verify %u %k %t
AuthorizedKeysCommandUser nobody

[23:43 opkuser@server ~]$
[23:43 opkuser@server ~]$
exit
Connection to server.example.com closed.
[opkuser@client ~]$
[opkuser@client ~]$
[opkuser@client ~]$ opkssh inspect ~/.ssh/id_ecdsa-cert.pub
--- SSH Certificate Information ---
Serial:            0
Type:              User Certificate
Key ID:            opkuser@example.com
Principals:        []
Valid After:       Not set
Valid Before:      Forever
Critical Options:  map[]
Extensions:
  permit-pty:
  permit-user-rc:
  openpubkey-pkt: [PKToken data] 1242 bytes
  permit-X11-forwarding:
  permit-agent-forwarding:
  permit-port-forwarding:

--- PKToken Structure ---
Payload:
{
  "aud": "opkssh",
```


ssh_lab — podman ‹ demo4_opkssh.sh — 127×35
.../ssh_lab — podman ‹ demo4_opkssh.sh
~/projects/ssh_lab — -zsh
~/projects/ssh_lab — -zsh
...lab — podman ‹ podman-compose up
~/projects/ssh_lab — -zsh

```
--- PKToken Structure ---
Payload:
{
  "aud": "opkssh",
  "azp": "opkssh",
  "email": "opkuser@example.com",
  "email_verified": true,
  "exp": 1769817475,
  "groups": [
    "allow_root",
    "important_user",
    "stepssh",
    "user"
  ],
  "iat": 1769816575,
  "iss": "https://idm.example.com:8443/oauth2/openid/opkssh",
  "jti": "55ea9208-16f4-4b03-9246-bbc00aeacd0c",
  "name": "OPK User",
  "nbf": 1769816575,
  "nonce": "5vEw72B9Hko65VEeV3gNh00SPuhPioH_7QSpC_ukSzc",
  "preferred_username": "opkuser@idm.example.com",
  "scopes": [
    "email",
    "openid",
    "profile"
  ],
  "sub": "9f508b80-ea7c-4954-bb30-4bb251d5001b"
}

--- Signature Information ---
Provider Signature (OP) exists
{
  "alg": "ES256",
  "kid": "eca6fbdd688a1d679a5e9dd184ea238e"
```


ssh_lab — podman ‹ demo4_opkssh.sh — 127×35
.../ssh_lab — podman ‹ demo4_opkssh.sh
~/projects/ssh_lab — -zsh
~/projects/ssh_lab — -zsh
...lab — podman ‹ podman-compose up
~/projects/ssh_lab — -zsh

```
--- Token Metadata ---
Issuer:             https://idm.example.com:8443/oauth2/openid/opkssh
Audience:           opkssh
Subject:            9f508b80-ea7c-4954-bb30-4bb251d5001b
Identity:           9f508b80-ea7c-4954-bb30-4bb251d5001b https://idm.example.com:8443/oauth2/openid/opkssh
Token Hash:         rSAmhL6h5Q2mQkgRHvqReOs-pRHKmbeI8lga_Y7hqro
Provider Algorithm: ES256
[opkuser@client ~]$
[opkuser@client ~]$
[opkuser@client ~]$ dig SSHFP server.example.com

; <<>> DiG 9.20.18 <<>> SSHFP server.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 10170
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 6, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;server.example.com.            IN      SSHFP

;; ANSWER SECTION:
server.example.com.     0       IN      SSHFP   4 2 F0F7C23A9086F9984E624466B15EFE1B243164A8C96A24F029D34FD7 90BBA6C5
server.example.com.     0       IN      SSHFP   4 1 CF880618201B8E97563DB8D5ACEFBC01D6EB3D89
server.example.com.     0       IN      SSHFP   3 2 8EDD585242A008BDBCCF591996B1523AF6F21DBECFB77B4680903DCA 5028EA0E
server.example.com.     0       IN      SSHFP   3 1 ECC60371F699E35BBAD9B4BFF1C078C2D9A9D81E
server.example.com.     0       IN      SSHFP   1 2 F9B1B0BABECBBB45BB1F5FEB6418B7BD27AED547B9DFC1DE74E8AA91 3D2DB9CD
server.example.com.     0       IN      SSHFP   1 1 64F20B2C19647FB91271BB920EAAB4011E7EB870

;; Query time: 2 msec
;; SERVER: 172.20.0.1#53(172.20.0.1) (UDP)
;; WHEN: Fri Jan 30 23:44:12 UTC 2026
;; MSG SIZE  rcvd: 287
```

Thanks for the fish!