GlobalLogic

A Hitachi Group Company

EDUCATION

Smart Start: Linux/Networking
Remote network access

Sergii Kudriavtsev

Agenda

* Networking: remote shell
* Networking: remote file systems
* Networking - GUI and 3rd party Remote Connectivity Software

Networking - remote shell

- Networking - remote shell
  - SSH
    - Server
      - sshd should be up and running
        - `$ service sshd status`
        - `$ service sshd start`
      - `/etc/ssh/sshd_config`
      - `$ man 5 sshd_config`
    - Client
      - `$ ssh user@box.example.com`
      - `$ ssh -o TCPKeepAlive=yes -o ServerAliveInterval=50 user@box.example.com`
      - Known hosts
        - `$HOME/.ssh/known_hosts`
      - Remote logging in using password
        - `$ ssh user@box.example.com`
          `Password:`

- Networking - remote shell
  - SSH Client
    - Remote logging in using keys
      - Generating keys
        - `$ ssh-keygen -t rsa -C "example.email@gmail.com" #leave passphrase empty for passwordless login (insecure)`
      - Protecting the keys
        - Private key should not be accessible by other users
      - Copying public key to remote host (secure channel must be used such as "ssh-copy-id", "ssh" or "scp" utilities)
        - `$ ssh-copy-id user@box.example.com`

          OR
        - `$ cat ~/.ssh/id_rsa.pub | ssh user@box.example.com "mkdir -p ~/.ssh && cat >> ~/.ssh/authorized_keys"`
      - Verifying login. If passphrase for the key was not empty, user will be asked for it
        - `$ ssh user@box.example.com`

- Networking - remote shell
  - SSH Client
    - Remote logging in using keys
      - Login without entering passphrase when passphrase **is not empty**. These steps to be performed on a client side (e.g. on a host which initiates connection).
        - Entering passphrase of the private key once per session (effective for current shell and its derived processes AND for other user's processes if SSH_AUTH_SOCK and SSH_AGENT_PID environment variables are set).
          - Add the following to your `.profile` or `.bash_profile`, so ssh-agent will be started automatically upon your login:
            ```
            eval `ssh-agent -s`
            ssh-add
            ```
          - Re-login or source your `.profile` or `.bash_profile`
            - `$ source ~/.profile`
              OR
            - `$ source ~/.bash_profile`
        - Check SSH connection, no passphrase should be requested:
          - `$ ssh user@box.example.com`
        - Stopping ssh-agent:
          - `$ eval `ssh-agent -k``

- Networking - remote shell - SSH Client
  - Remote logging in using keys
    - Entering passphrase of the private key once after system boot.
      - Install keychain (under root)
        - `$ sudo apt install keychain`
        - Add the following to your `.profile` or `.bash_profile`, so ssh-agent and gpg-agent will be started automatically upon your login. In case if ssh-agent and gpg-agent are not running, user will be asked for key's passphrase. Otherwise, user will not be asked and existing ssh-agent and gpg-agent processes will be used.
          `eval \`keychain --eval id_rsa\``
        - Re-login or source your `.profile` or `.bash_profile`
          - `$ source ~/.profile`
            OR
          - `$ source ~/.bash_profile`
        - Check SSH connection, no passphrase should be requested:
          - `$ ssh user@box.example.com`
        - Stopping ssh-agent and gpg-agent:
          - `$ keychain -k all`

- Networking - remote shell - SSH Client
  - Storing passphrase of the private key in plain text in a file, using it automatically
    - Install expect (under root): #expect - programmed dialogue with interactive programs.
      - `$ sudo apt install expect`
    - `$ which expect`
      `/usr/bin/expect`
    - Write a script for running ssh-add and supplying password to it from file:

```
$ cat > ~/ssh-add.expect << EOF
#!/usr/bin/expect -f
spawn ssh-add /home/ user/.ssh/id_rsa
expect "Enter passphrase for /home/ user/.ssh/id_rsa:"
send "mypassphrase\n";
interact
EOF
```

    - `$ chmod +x ~/ssh-add.expect`

- Networking - remote shell - SSH Client
  - Storing passphrase of the private key in plain text in a file, using it automatically

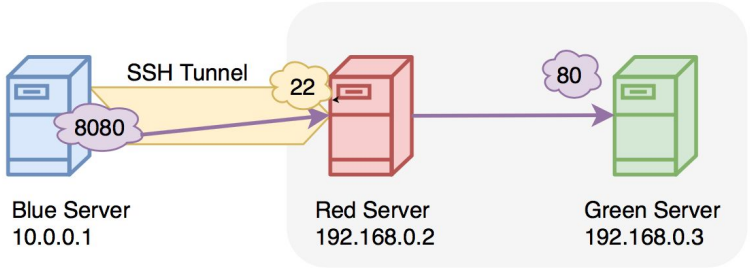
    …
    - Add the following to your `.profile` or `.bash_profile`, so ssh-agent will be started automatically upon your login:

```
if [ -z "$SSH_AUTH_SOCK" ]
then
        eval `ssh-agent -s`
        $HOME/ssh-add.expect
fi
```

      - Re-login or source your `.profile` or `.bash_profile`
        - `$ source ~/.profile`

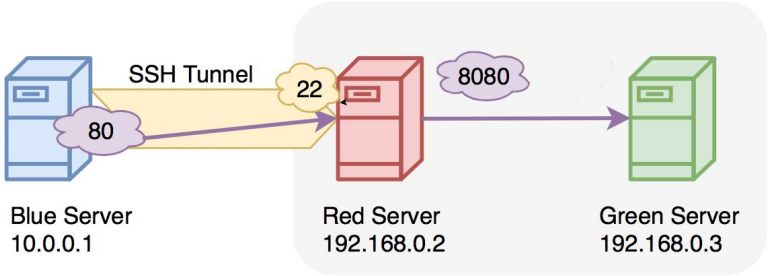
          OR
        - `$ source ~/.bash_profile`
      - Check SSH connection, no passphrase should be requested:
        - `$ ssh user@box.example.com`
    - Stopping ssh-agent:
        - `$ ssh-agent -k`

- Networking - remote shell - SSH Client



  - Tunneling
    - Remote Resources Accessible on Your Local System
    - `$ ssh -L [bind_address:]local_port:remote_host:remote_hostport` *`user@box.example.com`*
    - Examples:
      - `$ ssh -L 8080:192.168.0.3:80 user@10.0.0.2`
      - `$ ssh -L 55555:localhost:5432` *`user@box.example.com`*
      - `$ ssh -L localhost:55555:localhost:5432` *`user@box.example.com`*
      - `$ ssh -L 10.105.28.31:55555:localhost:5432` *`user@box.example.com`*
      - `$ ssh -L 55555:fbi.gov:80` *`user@box.example.com`*
      - `$ ssh -L 55555:fbi.gov:443` *`user@box.example.com`*

- Networking - remote shell - SSH Client



  - Tunneling
    - Local Resources Accessible on a Remote System
    - `$ ssh -R [bind_address:]remote_port:local_host:local_hostport user@box.example.com`
    - Examples:
      - `$ ssh -R 8080:localhost:80 user@10.0.0.2`
      - `$ ssh -R 55555:localhost:5432 user@box.example.com`
      - `$ ssh -R localhost:55555:localhost:5432 user@box.example.com`
      - `$ ssh -R 10.105.28.31:55555:localhost:5432 user@box.example.com`

- RSH - the legacy remote shell tool
  - rsh (obsolete) - The rsh (remote shell) program was a tool for remotely running a command on a remote computer. It has since been superceded by ssh.
    The rsh tool was introduced in BSD Unix in the 1980s. It was an important tool at the time, but it suffered from several shortcomings. Its security was poor, and its usability wasn't great.
  - Security Issues in rsh
    - IP addresses spoofing: **rsh** used **.rhosts** files and **/etc/hosts.equiv** for authentication. These methods relied on IP addresses and DNS for authentication.
  - Usability Issues in rsh
    - Additional manipulations are needed in order to open a terminal window and run arbitrary applications from the remote server (Setting DISPLAY variable, secure authentication tokens for X11 authentication, e.t.c.)

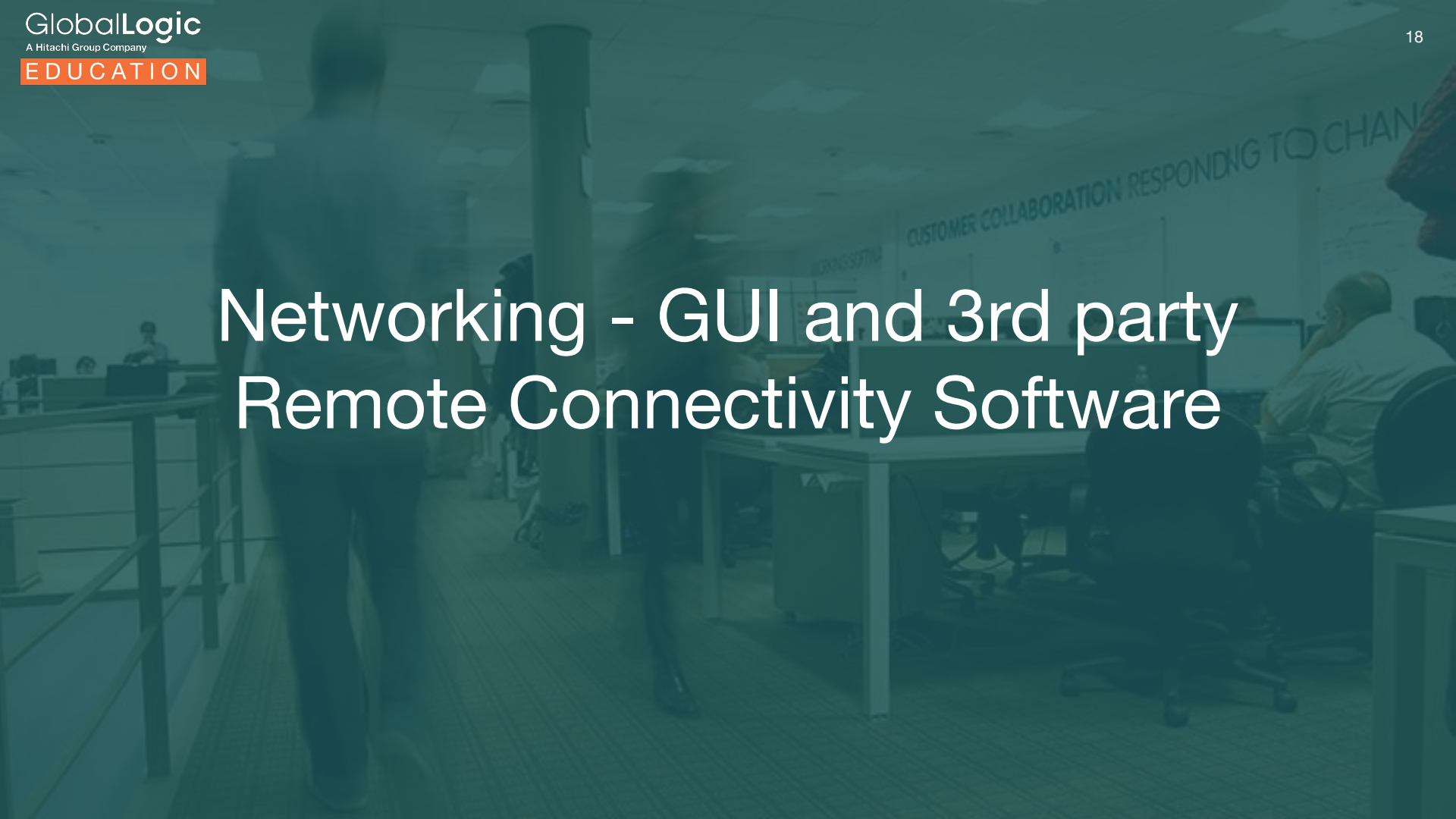
- Networking - remote shell - SSH Client
  - Telnet (obsolete)
    - `$ telnet remotehost remoteport`
    - Check if port is open on remote machine:
      - `$ telnet remotehost remoteport`
        - If cannot connect after a long period of time, press `Ctrl-C`
        - If connected and is not disconnected automatically:
          - press `Ctrl-]`
          - type "`quit`"

Networking - remote file systems

- Networking - remote file systems
  - NFS
    - Installation (under root)
      - `$ apt install nfs-kernel-server`
    - Server (under root)
      - Start NFS service
        - `$ systemctl enable nfs-server.service`
        - `$ systemctl start nfs-server.service`
      - `$ man 5 exports`
      - Configure `/etc/exports` (192.168.0.101 - client's IP):
        `/home 192.168.1.101(rw,sync,no_root_squash,no_subtree_check)`
      - `$ exportfs -a`
    - Client (192.168.0.100 - server's IP), (under root)
      - Mounting
        - `$ mount 192.168.1.100:/home /mnt/nfs/home`
      - Unmounting
        - `$ umount /mnt/nfs/home`

- Networking - remote file systems
  - SSHFS
    - Server
      - `$ service sshd start`
    - Client
      - Installation (under root)
        - `$ apt-get install sshfs`
      - Mounting (192.168.0.100 - server's IP)
        - `$ sshfs ironman@192.168.56.200:/home /mnt/sshfs/home`
      - Unmounting
        - `$ fusermount -u /mnt/sshfs/home`
        - `$ umount /mnt/sshfs/home`

- Networking - remote file systems
  - SAMBA
    - Client
      - Installation (under root)
        - `$ apt install samba-client smbclient samba-common cifs-utils`
      - Mounting (192.168.0.100 - server's IP)
        - (optional) `/etc/samba/smb.conf`
          - `workgroup = SYNAPSE`
        - `$ smbclient -U username -L IP`
        - `$ read -s PASSWD`
        - `$ mount -t cifs --verbose -o username='myuser',domain=SYNAPSE,password="$PASSWD",uid=1000,gid=1000 //192.168.0.100/home /mnt/cifs/home`
      - Unmounting
        - `$ umount /mnt/cifs/home`



Networking - GUI and 3rd party Remote Connectivity Software

- Networking - GUI Remote Connectivity
  - VNC

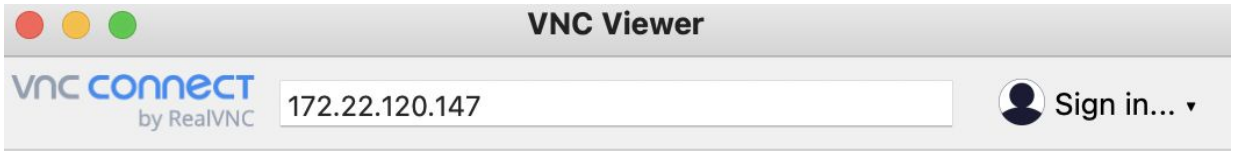
    

    - Installation (under root)
      - `$ apt install x11vnc`
    - Server
      - Start NFS service
        - `$ x11vnc`

          `…`

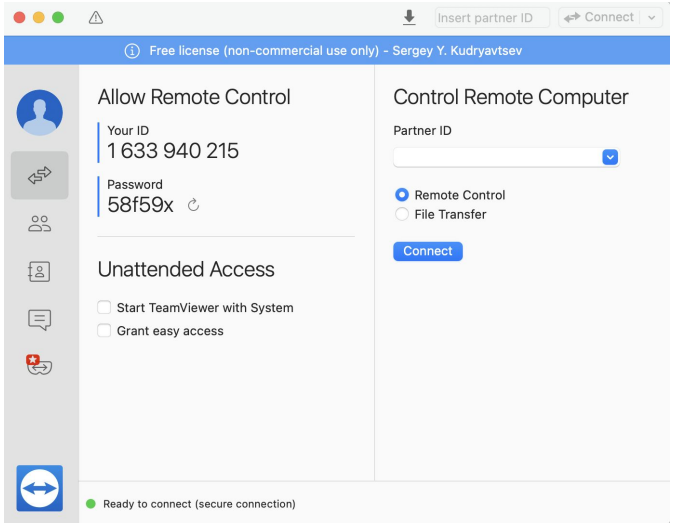
          `The VNC desktop is:  LWO1-LD-A26940.synapse.com:0`

          `PORT=5900`
    - Client (LWO1-LD-A26940.synapse.com - server's IP),
      - Connect: Create a new vnc connection using any of the available VNC Clients:



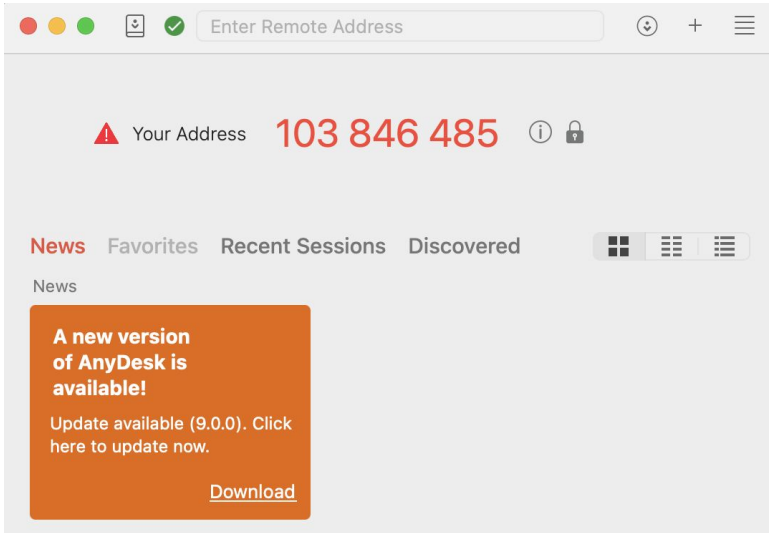
- Networking - GUI 3rd Party Remote Connectivity Software
  - TeamViewer
    - Installation
      - Download Latest free version from Web:

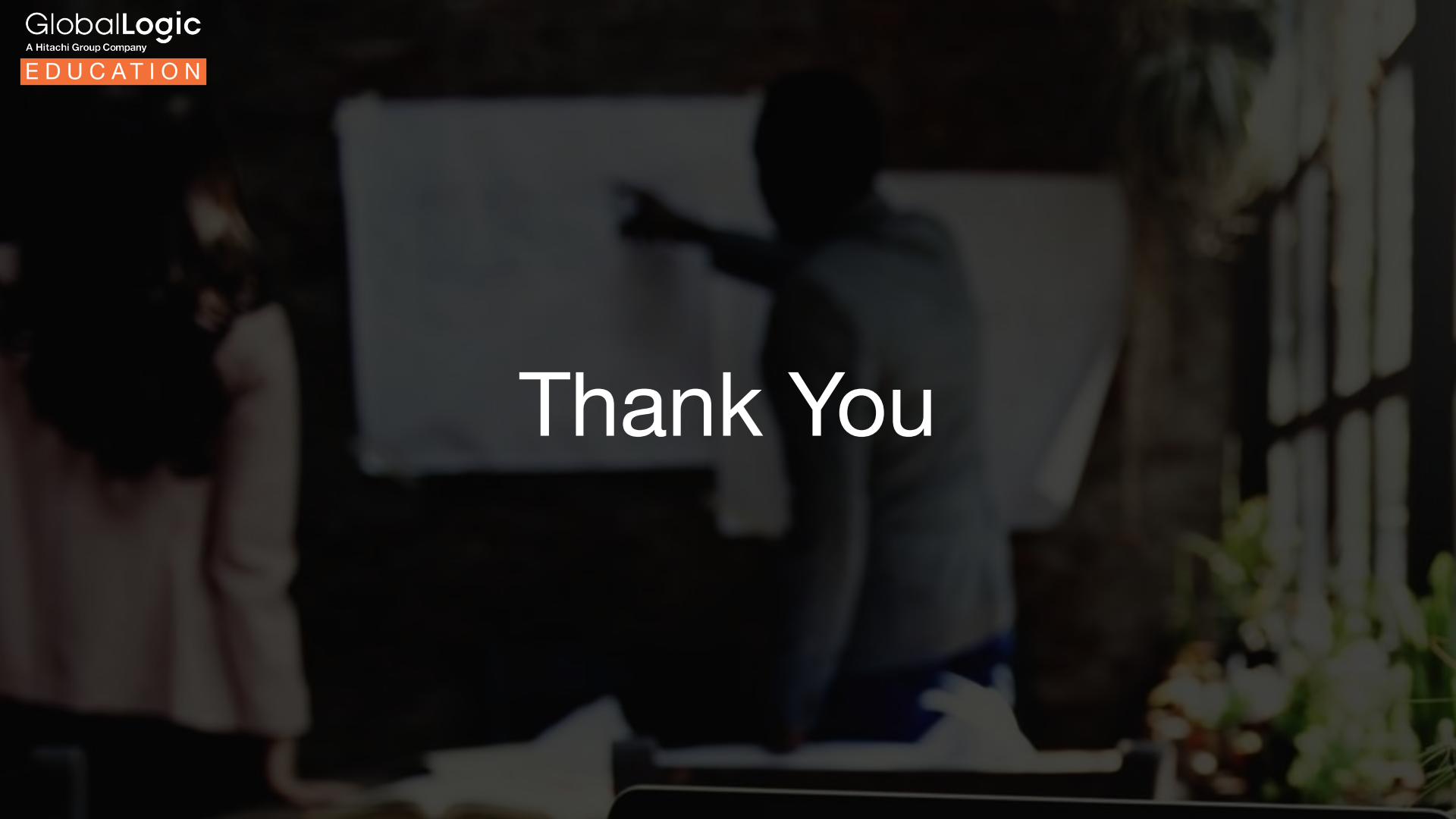
https://www.teamviewer.com/en/download/free-download-with-license-options/



- Networking - GUI 3rd Party Remote Connectivity Software
  - AnyDesk

    - Installation
      - Download Latest free version from Web: https://anydesk.com/en/downloads/




GlobalLogic
A Hitachi Group Company
EDUCATION
Thank You