# uInstall an Indy Node on Physical Hardware

Introduction: The following steps are one way to adhere to the Indy Node guidelines for installing an Indy Node on a Physical Hardware host. This Host can either be a VM or installed straight onto the physical server. For the hardware requirements applicable for your network, please refer to the steward technical requirements document or the network governance documents for your network. Keep in mind that there are multiple ways to set up networking and please feel free to follow your company's "best practices" when setting up your node as long as it meets the minimum requirements outlined in your GovernFance Documents.

1. Before you begin:
   1. For most governance frameworks' hardware requirements, you will need 2 NIC's and 2 subnets (one per NIC). Configure these before beginning the install.
   2. Hardware requirements usually include the following:
      1. 32 G RAM
      2. 8 CPU cores
      3. 1T RAIDed disk space
      4. 2 NICs with 2 Public IP addresses (1 per NIC)
   3. Create your own SSH key to use later for logging in to the Node.
      1. mkdir ~/pems
      2. ssh-keygen -P "" -t rsa -b 4096 -m pem -f ~/pems/validatornode.pem
2. Install Ubuntu 16.04 on the server (or VM).
   1. Note: I did not actually do this step when writing this guide, so there might be redundancies in future steps that I am not aware of.
   2. During installation, please make sure that the /var/lib/indy directory has about 1T of disk space available to it. It does not need to be that specific directory, it's okay if / has all of it.
3. Log in to your VM
   1. Use the admin user created during the installation process.
4. Configure networking
   1. From your instance's command prompt, run the command $ ip a and verify that you have 2 internal IP addresses that match what you have in your Node Installation Info spreadsheet. Note the names of the network interfaces. (mine were eth0 and eth1) The remaining instructions in this section assume eth0 is your original primary NIC (Client NIC) and eth1 is the secondary NIC (Node NIC).
   2. route -n
      1. Record the default gateway for later use. (e.g. mine is 10.1.1.1)
   3. sudo su -
   4. cd /etc/network/interfaces.d
   5. vim ../interfaces
      1. Cut the existing eth0 lines from this file in preparation for moving them to a new file in the current directory. (Mine only had 2 lines to remove.)
      2. Save the changed file.
      3. Example interfaces file now looks like:   
            
          auto lo  
          iface lo inet loopback
   6. Create 2 new network interface files using the following suggestions. These example files are configured so that an AWS instance can use 2 IP addresses on 2 different interfaces (it is preferred to have 2 subnets also, but if you do not, then you will need to adjust the following instructions to match that configuration).
   7. vim eth0.cfg (use <interface name>.cfg if your interface name is not eth0)
      1. For the following, substitute the Gateway and interface names recorded earlier for <Gateway> and <interface name> respectively. <interface name> should also match the name of the file it is in. Paste the eth0 lines cut from the interfaces file and then add the following lines, indented 3 spaces:  
           
          up ip route add <network>/24 dev eth0 src <local IP> tab 1

up ip route add default via <Gateway> dev <interface name> tab 1  
 up ip rule add from <local IP addr of <interface name>>/32 tab 1  
 up ip route add default via <Gateway> dev eth0  
 up ip route add <finish me>

* 1. up ip route add 10.1.1.0/24 dev eth0 src 10.1.1.17 tab 1
  2. Example eth0.cfg  
       
     auto eth0

iface eth0 inet dhcp

up ip route add default via 172.31.16.1 dev eth0 tab 1

up ip rule add from 172.31.26.65/32 tab 1

up ip rule add to 172.31.26.65/32 tab 1

up ip route flush cache

* 1. Repeat previous steps but for the second network interface. The simplest way to do that is probably:
     1. cp eth0.cfg eth1.cfg
     2. vi eth1.cfg
        1. Replace all instances of eth0 with eth1
        2. Change <local IP addr> to the one corresponding to eth1
        3. Change ‘tab 1’ to ‘tab 2’
        4. Change the IP addresses, gateways, and networks to the correct values
  2. Example eth1.cfg  
       
     auto eth1

iface eth1 inet dhcp

up ip route add default via 172.31.128.1 dev eth1 tab 2

up ip rule add from 172.31.128.42/32 tab 2

up ip rule add to 172.31.128.42/32 tab 2

up ip route flush cache

* 1. ifup eth1
     1. Check to make sure eth1 came up and is working properly using the command ip a. NOTE: When I ran ifup, I got the error "Failed to bring up eth1." because I had the wrong Gateway in eth1.cfg. At this point if you run route -n again, you will see the correct gateway and can adjust eth1.cfg to use it.
     2. If the eth0 interface becomes unusable, you should now be able to log in through eth1 to fix it (if you first allow ssh to the second NIC in your firewall).
  2. Restart your instance.
     1. reboot
  3. ssh to your instance again as described earlier.
     1. ssh <username>@<Client IP Address>

1. Restart the instance to check for NIC persistence.
   1. sudo reboot
   2. Login as before:
      1. ssh <username>@<Client IP Address>
   3. Check the NICs
      1. ip a
      2. The output of the above command should have 2 NICS with the correct IP addresses displayed.
      3. More NIC and disk verifications will occur during the Indy Node install process.
2. Configure the firewall
   1. If eth0 is the Client (9702) interface and eth1 is the NODE (9701) interface then the following will be the commands to setup the firewall to allow all client traffic on port 9702 and other nodes to communicate on port 9701 (the IP addresses are examples of other Nodes on the network).
   2. sudo ufw allow in on eth0 to any port 9702
   3. sudo ufw allow in on eth0 to any port ssh
   4. sudo ufw allow in on eth1 to any port ssh (to have a backup in case something fails on eth0, remove this later for highest safeguarding of Node port)
   5. sudo ufw allow in on eth1 from 3.135.134.42 to any port 9701
   6. sudo ufw allow in on eth1 from 3.17.215.226 to any port 9701
   7. sudo ufw allow in on eth1 from 50.18.84.131 to any port 9701/tcp
   8. sudo ufw show added (to double check what you have added)
   9. sudo ufw enable
3. Set up ssh key access capability for your own user:
   1. On your own workstation (the one that you will login to the node from) create a new ssh key:
      1. ssh-keygen -P "" -t rsa -b 4096 -m pem -f ~/pems/validatornode.pem
      2. cat ~/pems/validatornode.pem.pub
      3. copy results of above
   2. On Node machine login as yourself and then:
      1. sudo mkdir ~/.ssh
      2. vim ~/.ssh/authorized\_keys
      3. Paste the key copied earlier in this step.
4. Optional: Add a temporary administrative user as a safety net during Two Factor Authentication (2FA) setup.
   1. sudo adduser tempadmin
      1. You can safely ignore messages like “sent invalidate(passwd) request, exiting“
   2. sudo usermod -aG sudo tempadmin
   3. sudo mkdir /home/tempadmin/.ssh
   4. cp ~/.ssh/authorized\_keys /home/tempadmin/.ssh/authorized\_keys
   5. (Note: the previous step allows login as the tempadmin using the same ssh key that you created for yourself)
   6. Setup sshd\_config to temporarily allow password login for the tempadmin user.
      1. sudo vim /etc/ssh/sshd\_config
      2. Comment out the line containing ‘ChallengeResponseAuthentication’.
         1. #ChallengeResponseAuthentication no
      3. Make sure this line exists and is set to yes:
         1. PasswordAuthentication yes
      4. :wq to save and exit.
      5. sudo systemctl restart sshd
   7. The above lines will be altered again when you set up 2FA.
5. Setup 2FA for SSH access to the Node for your base user.
   1. Optional: Login in a separate terminal as your tempadmin user (that has sudo privileges) to have a backup just in case something goes wrong during setup.
      1. ssh tempadmin@<Client IP Addr>
   2. Install Google Authenticator, Duo, or Authy on your phone.
   3. As your base user on the Node VM, run the following to install the authenticator:
      1. sudo apt-get install libpam-google-authenticator
   4. Configure the authenticator to allow both password and SSH key login with 2FA by changing 2 files:
      1. sudo vim /etc/pam.d/common-auth
      2. Add the following line as the first uncommented line in the file
         1. auth sufficient pam\_google\_authenticator.so
         2. :wq
      3. sudo vim /etc/ssh/sshd\_config
         1. add/configure the following lines:
            1. ChallengeResponseAuthentication yes
            2. UsePAM yes
            3. AuthenticationMethods publickey,keyboard-interactive
            4. PasswordAuthentication no
         2. If you see any of the above lines commented out, remove the # to uncomment them. If you don't see any of the above lines, make sure to add them. If you see those lines configured in any different way, edit them to reflect the above.
         3. :wq
      4. sudo systemctl restart sshd
   5. Setup your base user to use 2FA by running the following from a terminal:
      1. google-authenticator
      2. Answer ‘y’ to all questions asked during the setup
      3. Save the secret key, verification code and scratch codes in a safe place. These are all just for your user and can be used to login or to recover as needed.
   6. On your phone app add an account and then scan the barcode or enter the 16 character secret key from the previous steps output.
   7. You should now be able to login using 2FA. First, check that login still works for your base user in a new terminal. If that doesn’t work, double check all of the configuration steps above and then restart sshd again. If it still doesn’t work, it’s possible that a server restart is required to make 2FA work (NOTE: It is dangerous to restart at this point, because then all of your backup terminals that are logged in will be logged out and there is a chance that you will lose access. Please check that all other steps have been executed properly before restarting.)
6. Add other administrative users:
   1. Send the other new admin users the following instructions for generating their own SSH keys:
      1. ssh-keygen -P "" -t rsa -b 4096 -m pem -f ~/pems/validatornode.pem
      2. Have the new users send you their public key (e.g. validatornode.pem.pub if they do the above command)
      3. Also have them send you their Public IP address so that you can add it to the local firewall to allow them access. Optionally, have them send a preferred username also.
   2. Add their IP addresses to the local firewall (if configured)l
      1. sudo ufw allow in on **eth0** from <ip-addr> to ssh
   3. Add the users to the server:
      1. Login to the Node as the base user.
      2. Run the following commands, substituting the username in for <newuser>
      3. sudo adduser <newuser>
         1. You can safely ignore messages like “sent invalidate(passwd) request, exiting“
      4. sudo usermod -aG sudo <newuser>
      5. Then create a file in the newusers home directory:
         1. sudo mkdir /home/<newuser>/.ssh
         2. sudo chown <newuser>:<newuser> /home/<newuser>/.ssh
         3. sudo vim /home/<newuser>/.ssh/authorized\_keys
         4. Paste the users public key into the open file and then save it (:wq)
         5. sudo chown <newuser>:<newuser> /home/<newuser>/.ssh/authorized\_keys
      6. Repeat the above for each new admin user you create.
   4. The new users are now able to login. Since 2FA is required, when you send the password to each of the new users, also send the following instructions (HINT: fill in the username, Client IP address, and password for them with the correct values):
      1. Thanks for agreeing to help with the administration of our Indy Validator Node. Please login to the node, change your password, and setup Two Factor Authentication (2FA) using the following instructions:
         1. ssh -i <your private SSH key file> <username>@<Client IP Addr>
         2. Type in <password> for your password
         3. On successful login, type in ‘passwd’ to change your password on the Validator Node. Please use a unique password of sufficient length and store it in a secure place (i.e. a password manager).
         4. To set up 2FA, type in ‘google-authenticator’
            1. Answer ‘y’ to all questions asked during the setup
            2. Save the secret key, verification code, and scratch codes in a safe place. These are all for your user and can be used to login or to recover as needed.
         5. Install Google Authenticator, Duo, Authy, or other google-authenticator compatible app on your phone or device.
         6. On your 2FA phone app, add an account, and then scan the barcode or enter the 16 character secret key from step 4’s output.
         7. Log out and then log back in to check and make sure it worked!
   5. All of your secondary admin users should be setup now.
7. You can now begin the Indy Node installation.