

Building scalable private cloud on a budget

How to use open source and automation to
build scalable private cloud on a budget



Who am I?


I am a DevOps Engineer with 7+ years of experience in Networks and Systems Engineering. I am passionate about computer networks, infrastructure and open source.

I work for Transnomis Solution, which helps governments with road information management and communication software. This talk presents private cloud running at Transnomis.



Why run your own infrastructure?

- **Cost** - Cloud services are expensive unless you are very small (with no staff to run infrastructure) or very large, with high scalability needs. Most companies have very predictable infrastructure loads.
Lot of cloud hosting companies have 30-40% FCF margin, even with high marketing and employee expenses.
- **Flexibility** - You can run exactly what you want and get granular level of control of your infrastructure. You may not need to self host everything. Some IaaS providers support good level of automation and you can just lease the hardware. Eg. Equinix Metal, OVH.
- **Open source** - Open source software means you don't need to build your own software.



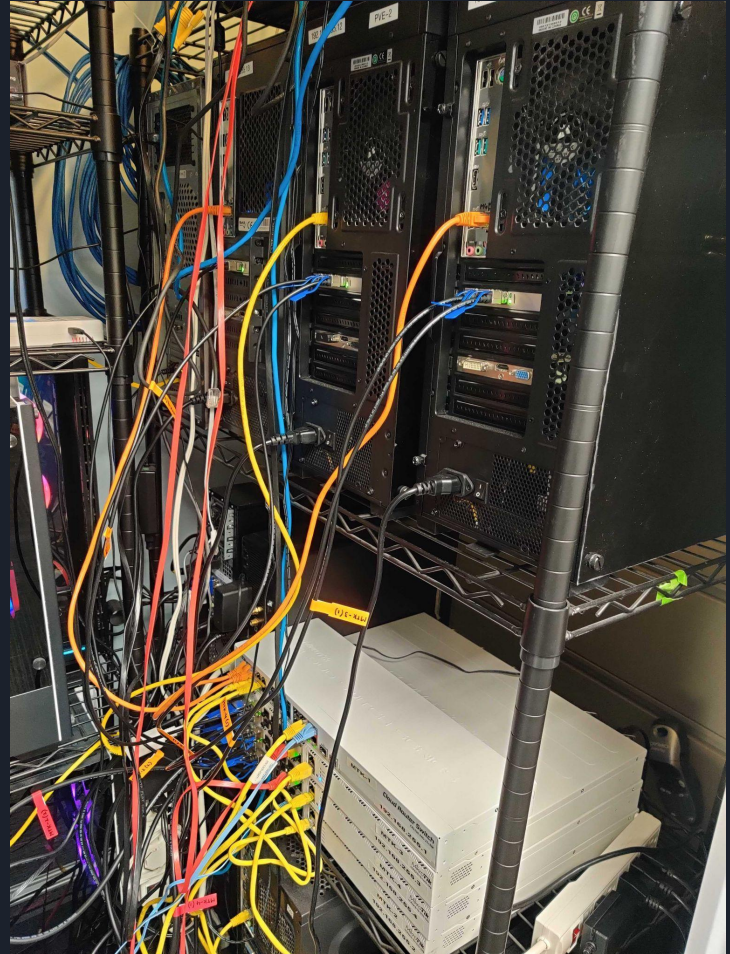
Topics we will be discussing - Building blocks for private cloud

- **VXLAN and OSPF** - For building scalable, isolated, highly redundant network.
- **Proxmox/Linux KVM** - for hypervisor.
- **Ceph** - For distributed storage.
- **Cloud-init/Cloudbase-init** - For automating Virtual Machine provisioning.
- **Ansible and Terraform** - For automation of VMs, network devices, hypervisors and VPNs.
- **LibreNMS** - For monitoring.
- **Hardware** - What to buy, sourcing etc.
- **Misc** - Identity management, VPNs.

Building the network

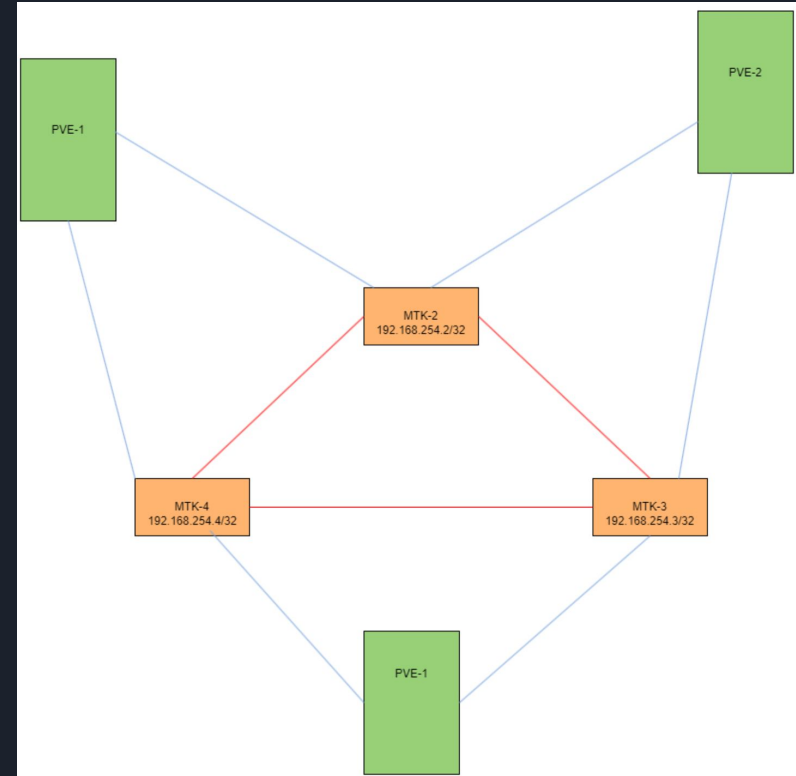
We want the network to be:

- **Scalable** - Should be able to scale to large number of physical machines.
- **Fast** - The network will run both the storage layer and other networks. We went with 25Gx2 per server as it was cheap enough to do.
- **Redundant** - Failure of multiple routers or network cables should not bring down the network.
- **Secure** - Networks should be isolated. VXLANs can provide VPC like functionality.



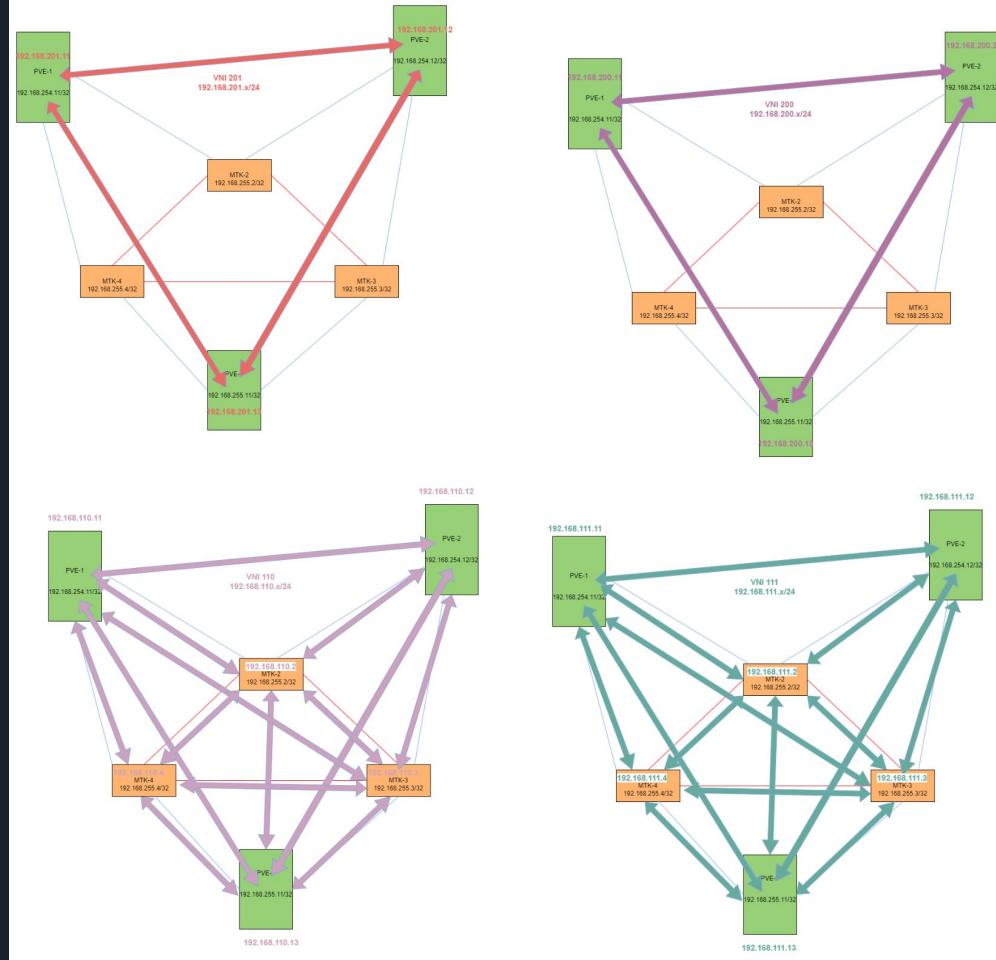
OSPF to maintain link state

- We use OSPF to maintain link state between all devices.
- Routers were connected over 100 Gbps backbone. For hypervisor hosts, they were connected to routers over 2x25 Gbps.
- OSPF makes all of devices connected work in a mesh and reroutes connections if we lose any devices or links.
- We use loopback IP on every device for them to communicate with each other.



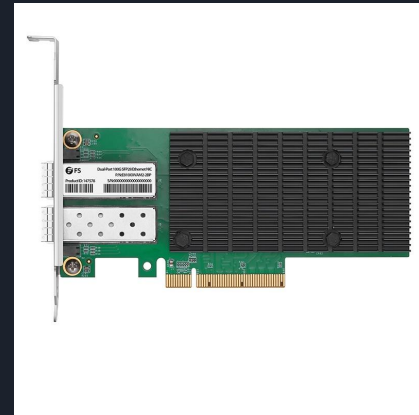
VXLAN over OSPF

- VXLANs can be thought of as layer 2 VPN tunnels over OSPF loopback IP, kind of like VPNs. They are similar to VPCs on other cloud providers.
- We run VXLANs between devices that need it.
- The top VXLAN tunnels is for Ceph and Proxmox Corosync clustering.
- The bottom two VXLAN tunnels is for VMs.
- VXLANs are scalable to 1000s of physical machines.
- For gateway we used VRRP.



Network budget

- We are using 3x **Mikrotik CCR2216-1G-12XS-2XQ** for core router. These have 12x 25 Gbps and 2x100 Gbps ports and can route packets at line rate. It's not perfect, but steal for the price. **Price: C\$3380.**
- We are using Intel E810 NIC cards, but the cheaper Mellanox ConnectX-4 are cheaper if you can find them. **Price: C\$330-580.**
- For cables DAC cables worked well for us. Fiber isn't necessary. **Price: C\$40-50.**



Automating network

- We used Ansible to automate the network.
- We created reusable roles for every network function.
- Terraform is better, but the module for Mikrotik is lacking lot of functionality.
- We then plugged it into Gitlab CI/CD pipeline for infrastructure.

```
TASK [firewall : Add firewall rules - custom (src address/dst address defined)] ***
changed: [MTK-3] => (item={'action': 'accept', 'chain': 'forward', 'src_address': '192.168.1.0/24', 'dst_address': '192.
changed: [MTK-4] => (item={'action': 'accept', 'chain': 'forward', 'src_address': '192.168.110.0/24', 'dst_address': '19
changed: [MTK-3] => (item={'action': 'accept', 'chain': 'forward', 'src_address': '192.168.110.0/24', 'dst_address': '19
changed: [MTK-4] => (item={'action': 'accept', 'chain': 'forward', 'src_address': '192.168.1.0/24', 'dst_address': '192.
changed: [MTK-3] => (item={'action': 'accept', 'chain': 'forward', 'src_address': '192.168.1.0/24', 'dst_address': '192.
changed: [MTK-4] => (item={'action': 'accept', 'chain': 'forward', 'src_address': '192.168.192.0/24', 'dst_address': '19
changed: [MTK-3] => (item={'action': 'accept', 'chain': 'forward', 'src_address': '192.168.192.0/24', 'dst_address': '19
TASK [firewall : Add firewall rules - DENY all] *****
changed: [MTK-4]

TASK [firewall : Include NAT] *****
included: /builds/infra/ansible-terraform/ansible/mikrotik/roles/firewall/tasks/nat.yml for MTK-4
TASK [firewall : Add firewall rules - DENY all] *****
changed: [MTK-3]

TASK [firewall : Include NAT] *****
included: /builds/infra/ansible-terraform/ansible/mikrotik/roles/firewall/tasks/nat.yml for MTK-3
TASK [firewall : Add src NAT rule] *****
changed: [MTK-4] => (item={'out_interface': 'bell', 'src_address': '192.168.110.0/23'})
TASK [firewall : Add src NAT rule] *****
changed: [MTK-3] => (item={'out_interface': 'bell', 'src_address': '192.168.110.0/23'})
TASK [firewall : Add src NAT rule] *****
changed: [MTK-4] => (item={'out_interface': 'bell', 'src_address': '192.168.143.1/32'})
TASK [firewall : Add src NAT rule] *****
changed: [MTK-3] => (item={'out_interface': 'bell', 'src_address': '192.168.143.1/32'})

PLAY RECAP *****
MTK-1                : ok=27  changed=23  unreachable=0    failed=0    skipped=25  rescued=0    ignored=0
MTK-2                : ok=34  changed=31  unreachable=0    failed=0    skipped=15  rescued=0    ignored=0
MTK-3                : ok=45  changed=42  unreachable=0    failed=0    skipped=4   rescued=0    ignored=0
MTK-4                : ok=45  changed=42  unreachable=0    failed=0    skipped=4   rescued=0    ignored=0

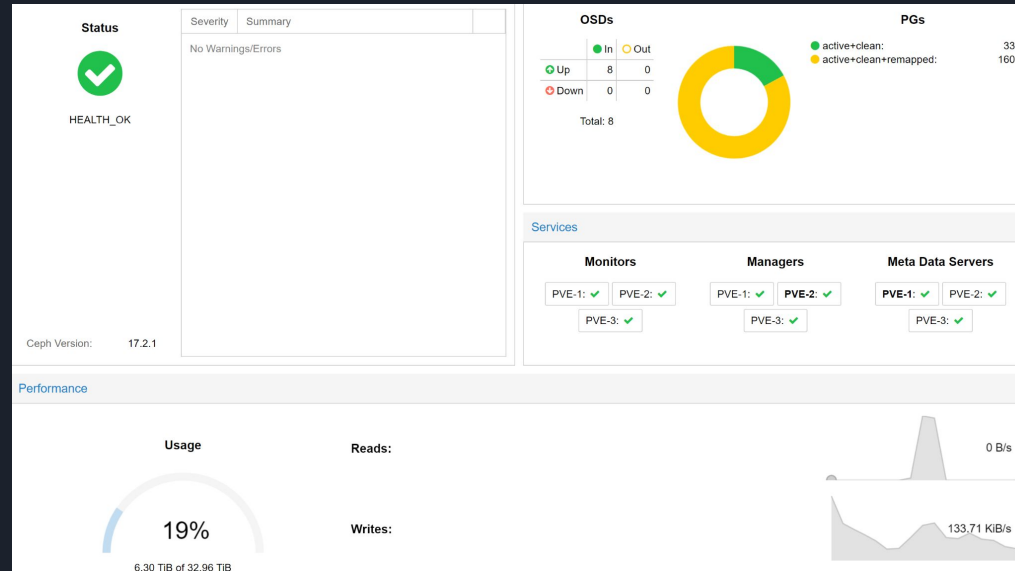
real    3m 58.00s
user    10m 3.50s
sys     0m 28.27s

Cleaning up project directory and file based variables
Job succeeded
```

```
111     dst_address: "192.168.192.0/24"
112     comment: "CUSTOM:8"
113 - action: "accept"
114   chain: "forward"
115   src_address: "192.168.192.0/24"
116   dst_address: "192.168.1.0/24"
117   comment: "CUSTOM:9"
118 interfaces:
119 - routeros_if: "ether1"
120   desc: "Management"
121   ipv4: "192.168.255.3/24"
122 - routeros_if: "qsf28-1-1"
123   desc: "OSPF ring interface"
124   ipv4: "192.168.141.2/30"
125   mtu: 9050
126   l2mtu: 9134
127 - routeros_if: "qsf28-2-1"
128   desc: "OSPF ring interface"
129   ipv4: "192.168.142.1/30"
130   mtu: 9050
131   l2mtu: 9134
132 - routeros_if: "lo"
133   desc: "loopback for OSPF"
134   ipv4: "192.168.254.3/32"
135 - routeros_if: "sfp28-1"
136   desc: "PVE-1 (Port 2)"
137   ipv4: "192.168.131.1/30"
138   mtu: 9050
139   l2mtu: 9134
140 - routeros_if: "sfp28-2"
141   desc: "PVE-2 (Port 2)"
142   ipv4: "192.168.132.1/30"
143   mtu: 9050
144   l2mtu: 9134
145 ospf:
146   router_id: "192.168.254.3"
147 interfaces:
148 - ospf_if: "qsf28-1-1"
149   type: "ptp"
150   auth: "md5"
151   auth_key: "transnomis"
152 - ospf_if: "qsf28-2-1"
153   type: "ptp"
154   auth: "md5"
155   auth_key: "transnomis"
156 - ospf_if: "lo"
157   type: "passive"
158 - ospf_if: "zerotier1"
159   type: "passive"
160 - ospf_if: "sfp28-1"
161   type: "broadcast"
162 - ospf_if: "sfp28-2"
163   type: "broadcast"
```

Proxmox+Ceph=❤️

- We use Proxmox on hypervisor hosts, which has excellent integration with Ceph distributed and redundant storage.
- Ceph can store Petabytes of data easily.
- Ceph and Proxmox Corosync clustering allows us to make VMs highly available and have live migration.
- Proxmox uses Linux KVM as hypervisor.
- Proxmox host post-provisioning is done with Ansible.



Terraform+Cloud-init+Ansible for Linux VMs

- We are using Terraform and Cloud-init for automating Linux VM creation. telmate/proxmox Terraform provider interacts with Proxmox VMs for VM creation and works very well.
- Cloud-init sets up the VM with correct hostname, DNS, timezone, adds users, sets up SSH keys.
- Ansible post provisions the VMs, adds it to monitoring, sets up correct applications.



Cloud-init config

- We create cloud-init templates that are pushed to every Proxmox host using Ansible. This tells Proxmox how to configure the VM, including host name, time zone.
- We use Jumpcloud for identity management, pushing users and SSH keys. We setup the agent with Cloud init.

```
1 #cloud-config
2 package_update: true
3 package_upgrade: true
4 timezone: America/Toronto
5 packages:
6   - sudo
7   - python3
8   - curl
9   - apt-rdepends
10  - apt-show-versions
11  - coreutils
12  - apt-curl
13  - dpkg
14  - grep
15  - hostname
16  - libc-bin
17  - libnss3
18  - libnss3-tools
19  - lsb-release
20  - lsof
21  - mawk
22  - openssl
23  - passwd
24  - procs
25  - sysvinit-utils
26  - tar
27
28 preserve_hostname: false
29 manage_etc_hosts: false
30 fqdn: {{ item.host }}.transnomis.com
31
32 write_files:
33   - path: /root/jc_register.sh
34     owner: root:root
35     permissions: '0755'
36     content: |
37       #!/bin/bash
38       sleep 120
39       conf="" cat /opt/jc/jcagent.conf""
40       regex="systemKey":\"(\\w+)\"""
41
42       if [[ $conf =~ $regex ]]; then
43         systemKey="${BASH_REMATCH[1]}"
44       fi
45
46       echo $$systemKey
47       API_KEY="{{ jumpcloud_api_key }}"
48       GROUP_ID="{{ jumpcloud_vm_groupid }}"
49       curl -X POST https://console.jumpcloud.com/api/v2/systemgroups/${GROUP_ID}/members \
50         -H "Accept: application/json" \
51         -H "Content-Type: application/json" \
52         -H "x-api-key: ${API_KEY}" \
53         -d@- <<EOF
54         {
55           "op": "add",
56           "type": "system",
57           "id": "$systemKey"
58         }
59       EOF
60
61 config_jc:
62   - &config_jc |
63     curl --tlsv1.2 --silent --show-error --header 'x-connect-key: {{ jumpcloud_vm_connectkey }}' https://kickstart.jumpcloud.com/kickstart | sudo bash
64
65 runcmd:
66   - [ sh, -c, *config_jc ]
67   - bash /root/jc_register.sh
68   - rm -rf /root/jc_register.sh
```

Terraform+Ansible for provisioning Linux VMs

- Terraform then uses Cloud-init config shown earlier to provision the VM.
- Ansible scripts then post provision the VM with specific software.
- You can run any application you like on VMs - Including Kubernetes, load balancer, S3, Docker containers on top of VMs.

```
prod.tf 1.54 KIB
1 resource "proxmox_vm_qemu" "prod-sql14-1" {
2   name       = "prod-sql14-1"
3   target_node = "PVE-2"
4   clone      = "ubuntu2204-cloudinit"
5   os_type    = "cloud-init"
6   nameserver = "192.168.111.21 192.168.111.22"
7   searchdomain = "transnomis.com"
8   cicustom   = "user=local:snippets/"
9   ipconfig0  = "ip=192.168.111.52/24"
10  hastate    = "started"
11  hgroup     = "ha_default"
12  onboot     = true
13  memory     = 8192
14  cores      = 4
15  agent      = 1
16  scsihw     = "virtio-scsi-pci"
17  disk {
18    size = "30G"
19    type = "scsi"
20    storage = "vm_data"
21    backup = 1
22  }
23  network {
24    model = "virtio"
25    bridge = "vbr111"
26    firewall = "false"
27  }
28  lifecycle {
29    ignore_changes = [

```

```
$ time ansible-playbook qa-sql14.yaml
[WARNING]: Invalid characters were found in group names but not replaced, use
-vvvv to see details
PLAY [qa-sql14] *****
TASK [Gathering Facts] *****
ok: [qa-sql14-1]
TASK [ansible.builtin.apt] *****
ok: [qa-sql14-1]
TASK [ansible-role-postgresql : include_tasks] *****
included: /builds/infra/ansible-terraform/ansible/vm/roles/ansible-role-postgresql/tasks/variables.yml for qa-sql14-1
TASK [ansible-role-postgresql : Include OS-specific variables (Debian).] *****
ok: [qa-sql14-1]
TASK [ansible-role-postgresql : Include OS-specific variables (RedHat).] *****
skipping: [qa-sql14-1]
TASK [ansible-role-postgresql : Include OS-specific variables (Amazon).] *****
skipping: [qa-sql14-1]
TASK [ansible-role-postgresql : Include OS-specific variables (Fedora).] *****
skipping: [qa-sql14-1]
TASK [ansible-role-postgresql : Define postgresql_packages.] *****
ok: [qa-sql14-1]
TASK [ansible-role-postgresql : Define postgresql_version.] *****
ok: [qa-sql14-1]
TASK [ansible-role-postgresql : Define postgresql_daemon.] *****
ok: [qa-sql14-1]
TASK [ansible-role-postgresql : Define postgresql_data_dir.] *****
ok: [qa-sql14-1]
TASK [ansible-role-postgresql : Define postgresql_bin_path.] *****
ok: [qa-sql14-1]
TASK [ansible-role-postgresql : Define postgresql_config_path.] *****
ok: [qa-sql14-1]
TASK [ansible-role-postgresql : Define postgresql_unix_socket_directories_mode.] ***
ok: [qa-sql14-1]
TASK [ansible-role-postgresql : Define postgresql_log_dir.] *****
ok: [qa-sql14-1]
TASK [ansible-role-postgresql : Define postgresql_effective_log_dir, if postgresql_log_dir is absolute] ***
skipping: [qa-sql14-1]
```

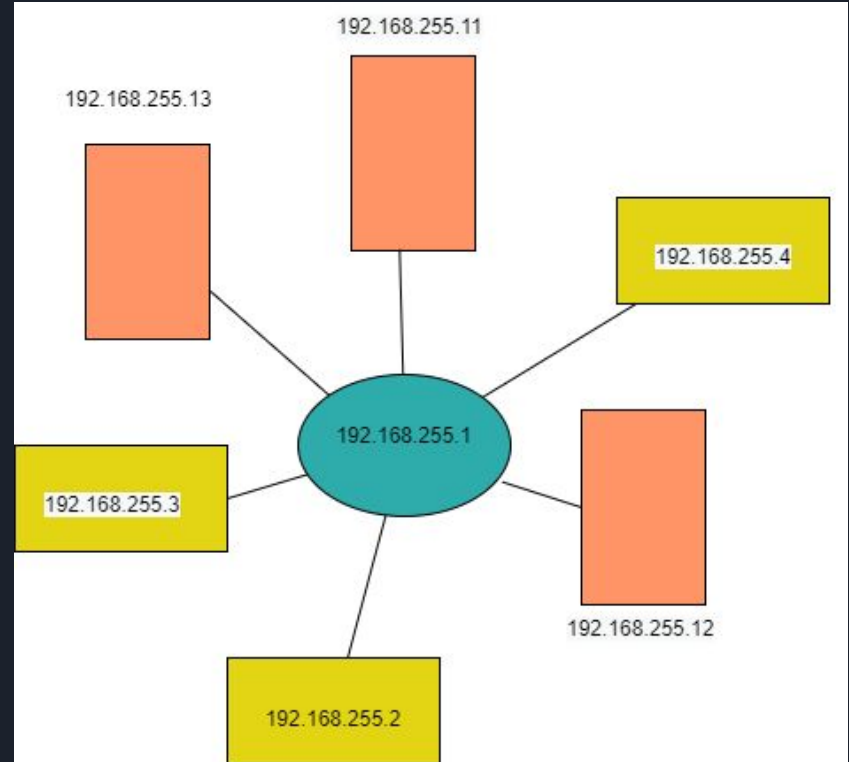
Cloudbase-init for provisioning Windows VMs

- Cloudbase-init does the same work as Cloud-init does on Linux.
- We can use Powershell to run scripts for the first time to ready the VM. We setup Jumpcloud this way.
- Terraform will use Cloudbase-init to provision the VM and Ansible(over WinRM) will post-provision it.

```
jumpcloud_windows_vm.j2 1.76 KB Edit Replace Delete
1 Content-Type: multipart/mixed; boundary="-----1598784645116016685--"
2 MIME-Version: 1.0
3
4 -----1598784645116016685--
5 Content-Type: text/cloud-config; charset="us-ascii"
6 MIME-Version: 1.0
7 Content-Transfer-Encoding: 7bit
8 Content-Disposition: attachment; filename="cloud-config"
9
10
11 set_timezone: America/Toronto
12 set_hostname: {{ item.host }}
13
14 -----1598784645116016685--
15 Content-Type: text/x-cfninitdata; charset="us-ascii"
16 MIME-Version: 1.0
17 Content-Transfer-Encoding: 7bit
18 Content-Disposition: attachment; filename="cfn-userdata"
19
20 #ps1
21
22 enable unrestricted policy
23 set-ExecutionPolicy Unrestricted
24
25 #disable administrator account
26 net user administrator /active:no
27
28 #wait for ping
29 do {
30   $ping = test-connection -comp 1.1.1.1 -count 1 -Quiet
31 } until ($ping)
32
33 #setup jumpcloud
34 cd $env:temp | Invoke-Expression; Invoke-RestMethod -Method Get -URI https://raw.githubusercontent.com/TheJumpCloud/support/master/scripts/windows/InstallWin
35
36
37 #setup XC
38 #variables
39 $api = "{{ jumpcloud_api_key }}"
40
41 #install nuget
42 Install-PackageProvider -Name NuGet -MinimumVersion 2.8.5.201 -Force
43
44 #import module
45 Import-Module -Name C:\windows\system32\WindowsPowerShell\v1.0\Modules\Microsoft.PowerShell.Management
46
47 #install XC
48 Install-Module -Name JumpCloud -Force
49 Connect-JSONline $api -force
50
51 #sleep
52 Start-sleep 120
53 #get system key
54 $test = select-string -Path $Env:ProgramFiles\JumpCloud\Plugins\Contrib\jccagent.conf -Pattern "systemkey":"([0-9a-fj-])" -AllMatches | X { $_.Matches } | X {
55   $test = $test.split(":")[1].Replace("","")
56 }
57
58 #connect to correct device group
59 Add-LocalSystemGroupMember -GroupName 'NewProxmoxVMs' -systemID $test
60
```


Monitoring and LibreNMS

- We use LibreNMS for monitoring hosts, services, network devices.
- We use separate network subnet to connect all devices together in separate network for out of band management.



Zerotier VPN

- We use Zerotier for VPN, which is a SDN VPN.
- Instead of traditional server-client model, every host is a node in the VPN cluster.
- We then can define, which network is behind which host and define network rules for every host.
- We have multiple sites and this is much easier than doing traditional VPNs.

The screenshot displays the Zerotier web interface for a device named 'varun-laptop-thinkpad'. The device's MAC address is 'e04ec2cc59' and its IP address is '192.168.192.253'. The device is managed by Terraform and is currently online. The configuration page includes several options:

- Exclude from SSO authentication: For a device which is a background or headless service, this option keeps it authorized on the network without expiration.
- Allow Ethernet Bridging: Bridging requires additional setup on the device. See manual and knowledgebase for more information. Mobile devices cannot be bridges.
- Do Not Auto-Assign IPs

Under the 'Capabilities' section, the following options are listed:

- admin
- externalsubnet
- gitlabssh
- onedriveshared
- prodmtottdb
- prodmtottrdp
- prodmtotstftp
- qadatabase
- reverseint

The 'Tags' section shows 'No tags defined.'



Thank you!