Email Domain Rewriting

How to add header rewriting to an O365 domain to support domain splitting & migration

The Problem

- i ❤ radio was reorganizing
  - needed to maintain their corporate access for a while
  - *also* needed to start sending email under their new domain name
  - Office 365 tenants on both end of this migration process
  - Office 365 supports multiple domain-name aliases, cool!
  - …but only for inbound mail.
  - Each user gets one primary domain and that gets used for outbound mail
  - A more flexible solution was needed!

The Proposed Solution

- Well, if Microsoft can't do it, surely UNIX can, right?
- Quick proof-of-concept using OpenBSD and OpenSMTPd, because it's super-small footprint, trivial to configure, and doesn't need hardening.
  - Success (sort of)!
    - OpenSMTPd is still too "lean" to support all the arbitrary rewriting we want.
    - So let's use Postfix instead, because I'm familiar with it.
  - Oh, and since we're running in the cloud, maybe use an OS that's supported by the cloud vendor
    - CentOS and/or Debian both provide adequate stability, longevity and supportability
  - And since this could go months/years before it gets re-used, a cheap cloud
    - So let's use OVH, instead of Azure or IBM or even AWS.
  - Needs to be easily scalable
  - Needs to be both documented and reproducible
    - So let's use Ansible for pretty much everything.

Solution Components

- VM(s) in OVH Public Cloud
  - Cost-effective. *Very* cost-effective compared to other options.
  - Multiple locations available in regions to (approximately) match O365 regions, for minimum latency
  - IPv4 + IPv6
  - Stable public IP addresses including control over PTRs for both v4 and v6
    - assuming that part of the management UI isn't broken today…
  - Authentication is not a Lovecraftian nightmare (like AWS)
    - admittedly, OVH's auth model is nowhere near flexible enough for more "*sensitive*" use cases
  - "Good enough!"

Debian vs. CentOS

- I honestly don't remember why…
- Minor advantages:
  - Longer projected support period
  - Larger selection of precompiled software
  - Bundled software is more up-to-date than CentOS (and all ELs)
  - Easier to override cloud-init for networking setup
    - cloud-init did some things in the wrong order and/or too late in the boot process for a dual-stack mail server. Might be OVH-specific, dunno.

Network Configuration

- Cloud providers universally use cloud-init.
- Cloud-init is great, except when it isn't.
- Luckily you can override it by creating one tiny file, then managing it like you normally would.
- …assuming your VM doesn't get new private IPs on every boot, anyway. In this case, having a LESS sophisticated cloud provider (OVH) worked out better.

```
root@lon:~# cat /etc/cloud/cloud.cfg.d/99-disable-network-config.cfg

network: {config: disabled}
```

```
root@lon:~# cat /etc/network/interfaces
### source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
# allow-hotplug ens4
# iface ens4 inet dhcp

auto ens3
iface ens3 inet static
        address 51.75.171.51/32
        gateway 51.75.171.1

iface ens3 inet6 static
        address 2001:41d0:801:2000::1dff/64
        gateway 2001:41d0:0801:2000:0000:0000:0000:0001
```

Ansible, for repeatability

- Here's the important Ansible files
  - AKA "playbooks"
  - Sorry about the formatting. Thank Microsoft Powerpoint.

```
root@lon:/etc/ansible/playbooks# cat all.yml
---
- hosts: all
  gather_facts: false

- name: SSH keys
  import_playbook: ssh-keys.yml

- name: APT packages
  import_playbook: packages.yml

- name: make network settings static
  import_playbook: network.yml

- name: force all Postfix variables
  import_playbook: postfix.yml

- name: sync servers
  import_playbook: sync.yml

# vim:set ts=2 sw=2 et nu ai si cursorcolumn:
```

```
root@lon:/etc/ansible/playbooks# cat packages.yml
---
- hosts: all
  gather_facts: false

  tasks:
  - name: pre-add Ansible repo key
    apt_key:
      keyserver: keyserver.ubuntu.com
      id: 93C4A3FD7BB9C367

  - name: configure APT repositories
    apt_repository:
      repo: "{{ item }}"
      state: present
    loop:
      - "deb http://debian.mirrors.ovh.net/debian stretch main contrib non-
free"
      - "deb-src http://debian.mirrors.ovh.net/debian stretch main contrib non-
free"
      - "deb http://security.debian.org/ stretch/updates main contrib non-free"
      - "deb-src http://security.debian.org/ stretch/updates main contrib non-
free"
      - "deb http://debian.mirrors.ovh.net/debian stretch-updates main contrib
non-free"
      - "deb-src http://debian.mirrors.ovh.net/debian stretch-updates main
contrib non-free"
      - "deb http://debian.mirrors.ovh.net/debian stretch-backports main
contrib non-free"
      - "deb-src http://debian.mirrors.ovh.net/debian stretch-backports main
contrib non-free"
      - "deb http://ppa.launchpad.net/ansible/ansible/ubuntu trusty main"
```

```
  - name: add packages
      package:
      name: "{{ packages }}"
      state: latest
    vars:
      packages:
        - ansible
        - apt-listchanges
        - bash-completion
        - bind9-host
        - binutils
        - bsd-mailx
        - certbot
        - cpp
        - curl
        - dnsutils
        - less
        - locales-all
        - locate
        - mtr-tiny
        - ncdu
        - needrestart
        - netcat-openbsd
        - netselect
        - netselect-apt
        - perl
        - postfix
        - postfix-doc
        - postfix-pcre
        - postfix-sqlite
        - psmisc
        - psutils
        - python-apt-common
        - python3-apt
        - rcs
        - swaks
        - unattended-upgrades
        - vim-nox
        - xz-utils
        - git
        - rsync

 # vim:set ts=2 sw=2 et nu ai si cursorcolumn:
```

```
root@lon:/etc/ansible/playbooks# cat all.yml
---
- hosts: all
  gather_facts: false

- name: SSH keys
  import_playbook: ssh-keys.yml

- name: APT packages
  import_playbook: packages.yml

- name: make network settings static ## still done by hand
  import_playbook: network.yml

- name: force all Postfix variables
  import_playbook: postfix.yml

- name: sync servers
  import_playbook: sync.yml

# vim:set ts=2 sw=2 et nu ai si cursorcolumn:
```

```
root@lon:/etc/ansible/playbooks# cat postfix.yml
---
- hosts: all
  gather_facts: true

  tasks:

    - name: unset /etc/mailname
      file:
        path: /etc/mailname
        state: absent

    - name: sync /etc/postfix directories
      synchronize:
        checksum: yes
        src: /etc/postfix/.
        dest: /etc/postfix/.

    - name: reset Postfix config completely
      command: truncate --size=0
/etc/postfix/main.cf
    - name: set Postfix options
      command:
      args:
        argv:
        - "postconf"
        - "-e"
        - "alias_maps=hash:/etc/aliases"
        - "compatibility_level=2"
        - "debug_peer_level=99"
        - "debug_peer_list=127.0.0.1, home.athompso.net, 204.16.144.114,
[2620:132:300e:700::]/64"
        - "lmtp_tls_security_level=may"
        - "local_header_rewrite_clients=static:all"
        - "mydestination=$myhostname, localhost.$mydomain, localhost,
$mydomain"
        - "mynetworks=127.0.0.0/8, cidr:/etc/postfix/allow_relay_by_ip"
        -
"sender_dependent_relayhost_maps=hash:/etc/postfix/per_sender_relayhosts"
        - "sender_canonical_maps=hash:/etc/postfix/translate_domains"
        - "smtpd_sender_restrictions=check_sender_access
hash:/etc/postfix/allowed_senders, reject"
        - "smtpd_tls_cert_file=/etc/letsencrypt/live/{{ ansible_hostname
}}.header-rewrite.net/fullchain.pem"
        - "smtpd_tls_key_file=/etc/letsencrypt/live/{{ ansible_hostname
}}.header-rewrite.net/privkey.pem"
        - "smtpd_tls_loglevel=1"
        - "smtpd_tls_received_header=yes"
        - "smtpd_tls_security_level=encrypt"
        - "smtp_tls_loglevel=1"
        - "smtp_tls_security_level=may"
        - "message_size_limit=153600000"

    - name: rebuild postfix databases
      command: postconf {{ item }}
      loop:
      - /etc/postfix/allowed_senders
      - /etc/postfix/per_sender_relayhosts
      - /etc/postfix/translate_domains

    - name: restart postfix
      service:
        name: postfix
        state: restarted

# vim:set ts=2 sw=2 et nu ai si cursorcolumn:
```

```
root@lon:/etc/ansible/playbooks# cat sync.yml
---
- hosts: all
  gather_facts: false

  tasks:

    - name: synchronize ansible (1st pass)
      synchronize:
        checksum: yes
        src: /etc/ansible/.
        dest: /etc/ansible/.

    - name: add changes to git
      local_action: command git add -A
      args:
        chdir: /etc/ansible
      run_once: true

    - name: check for changes to git
      local_action: command git diff-index --quiet HEAD
      run_once: true
      register: gitdiff
      failed_when: gitdiff.rc > 1

    - name: commit changes to git
      local_action: command git commit -a -m "Ansible automatic commit" -q
      args:
        chdir: /etc/ansible
      run_once: true
      when: gitdiff.rc == 1

    - name: synchronize ansible (2nd pass)
      synchronize:
        checksum: yes
        src: /etc/ansible/.
        dest: /etc/ansible/.

    - name: synchronize certbot
      synchronize:
        checksum: yes
        src: /etc/letsencrypt/.
        dest: /etc/letsencrypt/.


# vim:set ts=2 sw=2 et nu ai si cursorcolumn:
```

Ansible tricks 1

- “synchronize” module:
  - Src is always local to <machine ansible runs on>
  - Dst is always local to <machine ansible is targeting>
  - Uses rsync under the hood
- “local_action” module:
  - …is how you extend Ansible for one-off cases that don’t need their own entire new Ansible module
- “run_once” attribute:
  - Only run on ONE host no matter how many are in the targeted group.

Ansible tricks 2

- “command” module to set config values with postconf(8)
  - No need to use Ansible’s loop: construct because postconf(8) takes arbitrarily-long lists of variables and values on the command-line.
  - Could have used Ansible’s “flatten” approach but figuring that out was more work than using the “command” module.

“Sync” Process

- Unidirectional sync process is scalable
  - Assumes changes are only made on one system at a time.
    - This works for this system where only one person is making changes at a time. Will probably not work for multi-actor systems, YMMV.
- Main sync process only handles Ansible files
  - Keeps them in git, syncs git repo too
- Postfix playbook has its own rsync-only method
  - No git
  - Trivial to add, but wasn’t needed. (History of non-Ansible-controlled files there had very little value.)

Postfix translation 1

- Re-injection into O365
  - Required to handle DKIM, SPF, etc.
  - O365 leaves SMTP headers alone (mostly) when sender is authenticated
- “sender_dependent_relayhost”
  - Because O365 puts different tenants in different clusters, so you have to re-inject to the correct SMTP host.
- “ mynetworks= … , allow_relay_by_ip”
  - O365 has no way to do authenticated SMTP outbound. This is not a good solution, but it is an… *adequate* solution at this small scale. Both sender domain name and sending IP address must match, the attack scope is fairly small.

Postfix translation 2

- Sender_canonical_maps
  - This is the core function, and it's already built into Postfix!
  - Basically a search-and-replace function for sender email addresses.
  - Originally designed to translate internal-only hostnames into the canonical internet-visible email address.
    - e.g. "athompso@workstation.internal" → athompso@athompso.net
    - But can be used for arbitrary re-writing
- Allowed_senders
  - Trying to tighten up the security as much as possible
  - Allows entire domain names, not just individual email addresses

Geo-redundancy

- OVH has datacenters near Montreal, QC and London, England (plus many more)
  - That's separated enough to survive any physical event!
- Easy part: DNS
  - There's a two-valued A (and AAAA) record in the DNS zone; most MTAs will try each of the resolved IPs in some order.
  - We don't care about 30-sec timeouts, this is all M2M SMTP at this point.
- Harder part: TLS

TLS 1

- Postfix supports TLS, no big deal.
- Letsencrypt provides free certs that auto-renew, if you use CertBot to set them up.
- So what's the problem?
- Canonical names. SMTP client connects to "header-rewrite.net" but TLS gets a cert for a CN of "lon.header-rewrite.net". Oops.

TLS 2

- Solution: one cert with multiple valid hostnames!
- LE + Certbot lets you request a cert with multiple CNs. Tricky to automate, but still possible.
- Ultimately, we found a way to manually tell O365 to try this server, then that server.
  - Much easier than managing one “magic” cert!
  - O365 configuration is done by hand or by PowerShell, no big deal to specify two static server names!

Results

- My client (the sub-contractor) was able to offer a solution to *their* client (the contractor), who was able to offer a solution to *their* client (the actual users who needed this function).
- That intermediary (a Very Large Enterprise tech consultancy) has since *developed their own technology* to do this in a more Microsoft-ish way.
- Reportedly the project manager said: “This was the only piece of the entire project that worked properly the first time.”

Image credit: Charlie Cottrell, https://xeyeti.com/, https://charliecottrell.com/