Replicate your identity management

Jan Pazdziora, Red Hat

1. High availability of infrastructure services

When planning and deploying pieces of infrastructure that affect performance or core functionality of large number of machines and services in computer network, it is essential to aim for high availability, as well as for efficient use of resources. It is often helpful when the software itself provides admins with clear and easy way to view and manage the more complex topologies of redundant services.

FreeIPA project is an integrated umbrella for multiple identity management related solutions like directory server, Kerberos Key Distribution center (KDC), certificate system, DNS server, and others. It can be used to manage identities from users, hosts, or services to DNS records, and access control policies for Pluggable authentication modules (PAM) or sudo. Operation of client machines and services can be tightly dependent on the availability of the IPA server, even if the client-side System Security Services Daemon (SSSD) is built around caching and can alleviate some of the impact of unavailable server. Still, when new user logs in for the first time or system is accessed for the first time, their identities (via Name Services Switch (NSS) or DNS) need to be looked up and resolved, and authenticity and authorization of the user for given service verified. Keeping IPA server available and accessible without network latencies is thus important.

FreeIPA has been built around multi-master replication provided by the 389 Directory Server. The complexity of creating the replicas and managing the topology in alignment with overall network needs used to be a hurdle that might have left some deployments in less than optimal state. With FreeIPA releases 4.3 and 4.4, new features were introduced that make much simpler the setup of replicas, management of the replication topologies, as well as control over which servers the client hosts will prefer. Let's look at those features and use cases.

2. Setting up FreeIPA replica

With older FreeIPA versions, creating new replica server involved producing GPG-encrypted replica information file on the original master server using the Directory Manager password. This file then had to be copied manually to the replica machine and fed to the `ipa-replica-install` command, which again required the Directory Manager password. The need to run operations on the master together with use of credentials which were not otherwise used in standard day to day operation and management of IPA setup posed issues in automated scenarios and provisionings of replicas.

The current versions introduce the notion of promotion of existing IPA-enrolled client to replica, initiated from the client side, typically with admin's credentials. No operation is needed on the master, and gone is the manual copying of the replica information file.

In fact, even the admin's credentials are not needed on the replica machine. Special host group `ipaservers` is used to control the ability of machines to promote themselves to replicas, and the `ipa-replica-install` command can IPA-enroll machine itself, for example using one-time password for the host. If the admin (or some automated provisioning process) creates host entry for new replica, lets the IPA server generate random one-time password, and adds this new host to the `ipaservers` host group, the one-time password can be used on the replica machine to both IPA-enroll it and make it full replica.

One of the possible new workflows therefore can be:

```
client$ kinit admin
```

```
Password for admin@EXAMPLE.COM:

client$ ipa host-add replica.example.com --random
----------------------------
Added host "replica.example.com"
----------------------------
  Host name: replica.example.com
  Random password: ImgXN_VxNC,B
  Password: True
  Keytab: False
  Managed by: replica.example.com
client$ ipa hostgroup-add-member ipaservers --hosts=replica.example.com
  Host-group: ipaservers
  Description: IPA server hosts
  Member hosts: master.example.com, replica.example.com
------------------------
Number of members added 1
------------------------

replica# ipa-replica-install --password 'ImgXN_VxNC,B'
Configuring client side components
Client hostname: replica.example.com
Realm: EXAMPLE.COM
DNS Domain: example.com
IPA Server: master.example.com
BaseDN: dc=example,dc=com
...
Enrolled in IPA realm EXAMPLE.COM
Created /etc/ipa/default.conf
...
  Configuring directory server (dirsrv). Estimated time: 1 minute
  [1/43]: creating directory server user
  [2/43]: creating directory server instance
...
  [28/43]: setting up initial replication
Starting replication, please wait until this has completed.
Update in progress, 6 seconds elapsed
Update succeeded
  [29/43]: adding sasl mappings to the directory
...
  [2/2]: configuring ipa-otpd to start on boot
Done configuring ipa-otpd.
```

We see that the host entry and its host group membership was created from separate IPA-enrolled machine and no commands had to be invoked on the IPA master itself. On the replica machine, single command achieved both the IPA-enrollment, setup and configuration of the IPA server, as well as the replication agreement and configuration.

If the replica promotion starts with already IPA-enrolled machine, we can check that the original configuration which pointed to the master in `/etc/ipa/defaults.conf`

```
[global]
server = master.example.com
xmlrpc_uri = https://master.example.com/ipa/xml
```

gets updated to point to itself since the machine is now proper IPA server, after the promotion has finished:

```
xmlrpc_uri = https://replica.example.com/ipa/xml
```

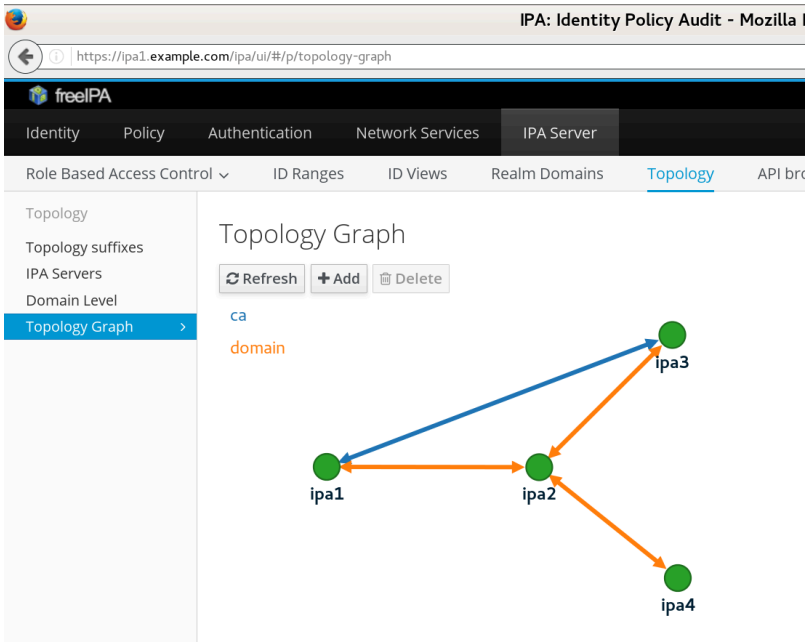
3. Replication topology management

When the first replica is created, it is obvious that there is only one replication agreement between the original master and the new replica. But in larger deployments when there are dozens of IPA server in multiple datacenters on different continents, the way how replication of changes is propagated and how replication agreements are structured becomes important. The more is not always the better here — as general rule, the number of replication agreements on any given IPA server should not exceed four.

In older versions, it was of course possible to manage the replication agreements but the command line tool `ipa-replica-manage` (and `ipa-csreplica-manage`, for the certificate system replication) had to be run on the IPA server, and the configuration of replication agreement was local to the pair of IPA servers in question.

With latest versions, all information about the replication topology, all the agreements (called segments), is stored in the directory server schema, and thus replicated to all servers in the domain. It is therefore possible to inspect the information with command line tools from IPA-enrolled clients or in WebUI. And the information is not just read-only view of the situation — when segment is created in the directory server, that information is replicated to all servers and when it reaches the target nodes of the new segment, establishment of new replication agreement is triggered. This is especially important when large domain of IPA servers needs to be managed centrally, without the ability to directly interact (with SSH) with the target IPA machines that might be in completely different geographical location. Having a way for replication to reach the remote IPA servers (potentially via chain of other IPA replicas) is enough to setup the replication agreement.

Let us assume situation of four IPA servers. The existing topology can be displayed graphically in Topology Graph in WebUI



or listed with `topologysegment-find` command:

```
ipa1$ ipa topologysuffix-find
---------------------------
2 topology suffixes matched
---------------------------
  Suffix name: ca
  Managed LDAP suffix DN: o=ipaca
```

```
  Suffix name: domain
  Managed LDAP suffix DN: dc=example,dc=test
----------------------------
Number of entries returned 2
----------------------------
ipa1$ ipa topologysegment-find domain
------------------
3 segments matched
------------------
  Segment name: ipa1.example.com-to-ipa2.example.com
  Left node: ipa1.example.com
  Right node: ipa2.example.com
  Connectivity: both

  Segment name: ipa2.example.com-to-ipa3.example.com
  Left node: ipa2.example.com
  Right node: ipa3.example.com
  Connectivity: both

  Segment name: ipa2.example.com-to-ipa4.example.com
  Left node: ipa2.example.com
  Right node: ipa4.example.com
  Connectivity: both
----------------------------
Number of entries returned 3
----------------------------
ipa1$ ipa topologysegment-find ca
-----------------
1 segment matched
-----------------
  Segment name: ipa1.example.com-to-ipa3.example.com
  Left node: ipa1.example.com
  Right node: ipa3.example.com
  Connectivity: both
----------------------------
Number of entries returned 1
----------------------------
```

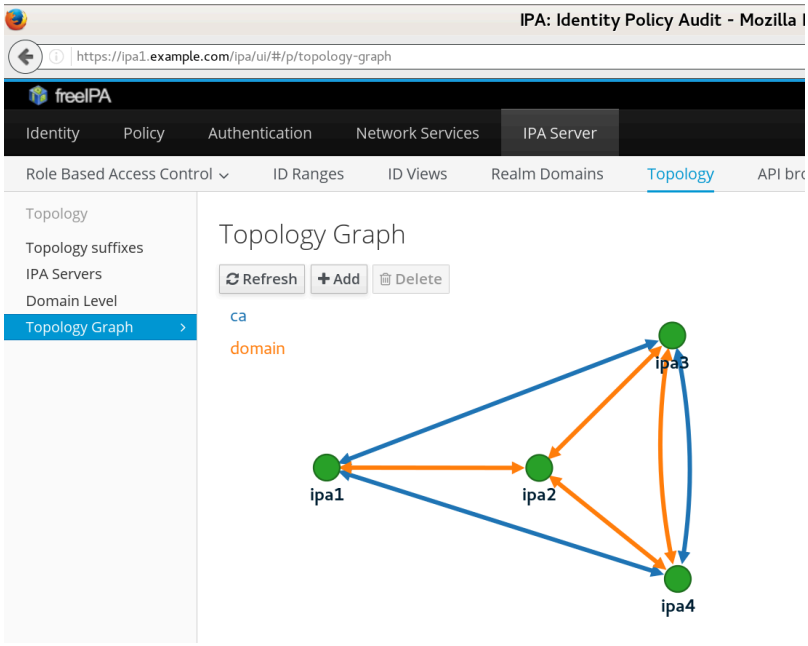
Note that there are actually two replication topologies here — the *domain* for the IPA domain information and *ca* for the certificate system data. We see that only **ipa1** and **ipa3** run the certificate servers and have direct replication segment for that purpose, while the domain data is replicated via a “central” node **ipa2**.

If we'd like to add new segment (replication agreement) between servers **ipa3** and **ipa4**, we can do so from any server, via WebUI or via command line tool. On the command line, we could use

```
ipa1$ ipa topologysegment-add domain ipa3.example.com-to-ipa4.example.com \
    --leftnode=ipa3.example.com --rightnode=ipa4.example.com
----------------------------------------------------
Added segment "ipa3.example.com-to-ipa4.example.com"
----------------------------------------------------
  Segment name: ipa3.example.com-to-ipa4.example.com
  Left node: ipa3.example.com
  Right node: ipa4.example.com
  Connectivity: both
```

If we refresh the Topology Graph on the WebUI, the new line between **ipa3** and **ipa4** will be shown there as well.

It is worth noting that segment can be added only between nodes that already act in given role. For instance, if we want to add replication agreement between **ipa3** and **ipa4** for the *ca* suffix for the certificate system, we first have to enable and configure the certificate system server on **ipa4** with `ipa-ca-install`. That will establish replication agreement with the currently configured CA host (**ipa1**) and only then we can add the next segment. Running `ipa topologysegment-find` or refreshing the page with the Topology Graph in WebUI would confirm our new replication topology:



4. DNS-based locations

So far we have focused on replica setup and management on the server side. We can easily create replicas and tune replication topology. But how are the client machines able to use the replicated setup?

In typical scenarios, IPA-enrolled Linux machines will use `ipa-client-install` to configure various components of the operating system. That configuration can explicitly name a particular IPA server, or it can rely on DNS SRV records to be used to discover the correct service endpoint to use. In case of SSSD, the `ipa_server` can force the SRV lookup with `_srv_` token, and it can be mixed with hostname as well:

```
[domain/example.com]
ipa_server = ipa1.example.com, _srv_
```

will prioritize `ipa1.example.com` and fallback to any other server is can find via DNS. In older versions of FreeIPA, this was the primary mechanism available for “pinning” IPA clients to a particular close (in terms of smallest latency) server.

The problem with this solution is its client-side nature. If additional IPA server is installed in local datacenter, all clients might suddenly need the new server listed in addition to the existing `ipa1.example.com`:

```
[domain/example.com]
ipa_server = ipa1.example.com, ipa2.example.com, _srv_
```

The order of those hostnames would likely mean that the first server will be used by vast majority of clients while the second one will only be utilized when clients cannot reach the first one — hardly a fair load balancing. Other services and libraries might not even support mixing SRV DNS mechanism with explicit hostnames, making localized behaviour setup even harder.

To solve this problem, FreeIPA 4.4 introduces DNS-based locations. They allow grouping of IPA servers and assigning priorities to them, per individual locations.

The basic expectation is that in every network subset where clients are supposed to primarily use certain set of IPA servers, there is at least one IPA server (replica) running embedded DNS server, and that clients in that network subset are configured to use that DNS server. This part of the configuration is outside of IPA's handling — typically DHCP in individual network segments will be configured to provide local DNS server IP address first, and it will either be directly the IPA DNS server, or local DNS server which queries the IPA DNS server during recursive resolution.

IPA DNS server which is assigned to certain location (for example `emea`) will autogenerate location-based CNAME records for certain names: `_kerberos._udp.example.com`, `_kerberos._tcp.example.com`, `_ldap._tcp.example.com`, ... they all will return respective name under `emea._locations.example.com`. Those SRV records will in turn contain priorities for individual IPA servers based on their membership in given location, and weights entered by admin.

Let us see a simple `emea` location configuration:

```
$ ipa location-show emea
  Location name: emea
  Servers: ipa1.uk.example.com, ipa2.uk.example.com
  Advertised by servers: ipa1.uk.example.com, ipa2.uk.example.com
  Servers details:
    Server name: ipa1.uk.example.com
    Service weight: 10
    Service relative weight: 25.0%
    Enabled server roles: CA server, DNS server, NTP server

    Server name: ipa2.uk.example.com
    Service weight: 30
    Service relative weight: 75.0%
    Enabled server roles: DNS server, NTP server
```

When client machine resolves SRV record to autodiscover a service, it will be “redirected” via CNAME answer to the location-specific SRV record with location-specific priorities.

```
$ dig +short @ipa1.uk.example.com. _kerberos._tcp.example.com SRV
_kerberos._tcp.emea._locations.example.com.
0 10 88 ipa1.uk.example.com.
0 30 88 ipa2.uk.example.com.
50 10 88 ipa1.houston.example.com.
$ dig +short @ipa1.houston.example.com. _kerberos._tcp.example.com SRV
_kerberos._tcp.us._locations.example.com.
50 10 88 ipa1.uk.example.com.
0 10 88 ipa1.houston.example.com.
50 30 88 ipa2.uk.example.com.
```

We see that depending on which IPA DNS server we query, we get priorities 0 for servers in the same location and 50 for servers outside of it.

This way, SSSD only needs to use `ipa_server = _srv_` and other components on the IPA client machines do not need to configure explicit hostnames of servers and services of IPA domain. Any modification to the priorities takes place on the server side, and is of course replicated to all IPA servers. This approach also caters nicely to the needs of roaming users within the organization. Depending on where they connect their laptop to the network, they will obtain local DNS server IP address, and will thus resolve the SRV records via the closest location.

5. Conclusion

With features focused on replicated setups and their management, latest FreeIPA releases make large-scale identity management deployments easier and more efficient. There are no longer any excuses for not having well set up FreeIPA redundancy.

References

FreeIPA project. https://www.freeipa.org/

Replica promotion. https://www.freeipa.org/page/V4/Replica_Promotion

Managing replication topology. https://www.freeipa.org/page/V4/Manage_replication_topology

DNS locations. https://www.freeipa.org/page/V4/DNS_Location_Mechanism