
# Men&Mice Micetro Collection

**Ton Kersten**

# Table of Contents

# Chapter 1. Ansible setup for Men&Mice Micetro

With the Ansible setup for Men&Mice Micetro you can manage a Men&Mice installation through Ansible. The Ansible modules and plugins connect to the Men&Mice Micetro API and perform all needed actions.

The modules and plugins need to be installed on the Ansible control node, often called the Ansible Master and Ansible needs to be configured so that the modules and plugins can be found by Ansible.

## 1.1. Installation

Installing the Ansible modules and plugins is a straight forward process, just install from the Ansible Galaxy.

```
ansible-galaxy collection install ansilabnl.micetro
```

### 1.1.1. Requirements

The Ansible integration modules and plugins do not need anything beyond a standard Ansible installation. The minimum Ansible version is 2.9 and up and the required Python version is 3.6+.

  

# Chapter 2. Ansible plugins

## 2.1. ansilabnl.micetro.freeip plugin

This Men&Mice FreeIP lookup plugin finds one or more free IP addresses in a certain network, defined in Men&Mice Micetro.

### 2.1.1. Options

- `claim`: Claim the IP address(es) for the specified amount of time in seconds
- `excludedhcp`: exclude DHCP reserved ranges from result
- `filter`: Men&Mice Micetro filter statement. Filter validation is done by the Men&Mice Micetro, not in the plugin. More filter info on `https://docs.menandmice.com/display/MM930/Quickfilter`
- `multi`: Get a list of x number of free IP addresses from the requested zones.
- `network`: (required) Network zone(s) from which the first free IP address is to be found. This is either a single network or a list of networks
- `ping`: ping the address found before returning.
- `provider`: (required) Definition of the Men&Mice Micetro API provider.

### 2.1.2. Usage

When using the Men&Mice FreeIP plugin something needs to be taken into account. When running an Ansible lookup plugin, this lookup action takes place every time the variable is referenced. So it will not be possible to claim an IP address for further reference, this way. This has to do with the way Ansible works. A solution for this is to assign all collected IP addresses to an Ansible fact, but here you need to make sure the factname is not used over multiple hosts.

Example usage:

  

*Listing 1. Claim IP addresses in one or more ranges*

```
---
- name: Men&Mice FreeIP test play
  hosts: localhost
  connection: local
  become: false

  vars:
    provider:
      mmurl: http://mmsuite.example.net
      user: apiuser
      password: apipassword
    network: examplenet

  tasks:
    - name: Set free IP addresses as a fact
      set_fact:
        freeips: "{{ query('ansilabnl.micetro.freeip',
                           provider,
                           network,
                           multi=15,
                           claim=60,
                           startaddress='192.168.63.100',
                           excludedhcp=True,
                           ping=True)
                  }}"

    - name: Get the free IP address and show info
      debug:
        msg:
          - "Free IPs          : {{ freeips }}"
          - "Queried network   : {{ network }}"
          - "Ansible version   : {{ ansible_version.full }}"
          - "Python version    : {{ ansible_facts['python_version'] }}"
          - "Python executable : {{ ansible_facts['python']['executable'] }}"

    - name: Loop over IP addresses
      debug:
        msg:
          - "Next free IP      : {{ item }}"
      loop: "{{ freeips }}"
```

  

```
# ansible-playbook mmtest.yml

PLAY [Men&Mice FreeIP test play] ************************************

TASK [Gathering Facts] **********************************************
ok: [localhost]

TASK [Set free IP addresses as a fact] ******************************
ok: [localhost]

TASK [Get the free IP address and show info] ************************
ok: [localhost] => {
    "msg": [
        "Free IPs          : ['192.168.63.203', '192.168.63.204']",
        "Queried network   : nononet",
        "Ansible version   : 2.9.7",
        "Python version    : 3.6.8",
        "Python executable : /usr/libexec/platform-python"
    ]
}

TASK [Loop over IP addresses] ***************************************
ok: [localhost] => (item=192.168.63.203) => {
    "msg": [
        "Next free IP      : 192.168.63.203"
    ]
}
ok: [localhost] => (item=192.168.63.204) => {
    "msg": [
        "Next free IP      : 192.168.63.204"
    ]
}

PLAY RECAP **********************************************************
localhost : ok=4  changed=0  unreachable=0  failed=0  skipped=0  rescued=0
ignored=0
```

## 2.2. ansilabnl.micetro.inventory plugin

This plugin generates the inventory from Men&Mice Micetro. It supports reading configuration from both a YAML configuration file and environment variables. If reading from the YAML file, the filename must end with `ansilabnl.micetro.inventory.(yml|yaml)`, the path in the command would be `/path/to/ansilabnl.micetro.inventory.(yml|yaml)`. If some arguments in the configuration file are missing, this plugin will try to fill in the missing arguments by reading from environment variables. If reading configurations from environment variables, the path in the command must be `@ansilabnl.micetro.inventory`.

Valid configuration filenames are:

- `ansilabnl.micetro.inventory`
- `mmsuite`
- `mandm`

- menandmice
- mandmsuite
- ansilabnl.micetro.suite
- mandm_suite

### 2.2.1. Options

There are two sets of configuration options, the options for the inventory plugin to function correctly and for Ansible to know how to use the plugin.

#### Plugin configuration

The ansilabnl.micetro.inventory plugin is configured through a configuration file, named ansilabnl.micetro.inventory.yml and the options are:

- plugin: Name of the plugin (ansilabnl.micetro.inventory)
- host: Men&Mice Micetro to connect to (http://mmsuite.example.net)
- user: UserID to connect with (apiuser)
- password: The password to connect with (apipasswd)
- filters: Filter on custom properties, can be more than 1 and should be a list. If multiple filters are given, they act as an **and** function
- ranges: What IP ranges to examine (172.16.17.0/24) Multiple ranges can be given, they act as an **or** function

When both *ranges* and *filters* are supplied that will result in an **and** function.

Example:

```
filters:
  - location: home
  - owner: tonk
ranges:
  - 192.168.4.0/24
  - 172.16.17.0/24
```

Would result in an inventory for all host that have the location: home **and** owner: tonk custom properties set **and** are either a member of the 192.168.4.0/24 **or** 172.16.17.0/24 range.

An example of the ansilabnl.micetro.inventory.yml file:

```
plugin: ansilabnl.micetro.inventory
host: "http://mmsuite.example.net"
user: apiuser
password: apipasswd
filters:
  - location: London
ranges:
  - 172.16.17.0/24
```

  

#### Environment variables:

The `ansilabnl.micetro.inventory` plugin can also be configured through environment variables

```
export MICETRO_HOST=YOUR_MICETRO_HOST_ADDRESS
export MICETRO_USER=YOUR_MICETRO_USER
export MICETRO_PASSWORD=YOUR_MICETRO_PASSWORD
export MICETRO_FILTERS=YOUR_MICETRO_FILTERS
export MICETRO_RANGES=YOUR_MICETRO_RANGES
```

When reading configuration from the environment, the inventory path must always be `@ansilabnl.micetro.inventory`.

```
ansible-inventory -i @ansilabnl.micetro.inventory --list
```

#### Ansible configuration

Ansible needs to know about the `ansilabnl.micetro.inventory` plugin and also has some extra configuration options. First the `ansilabnl.micetro.inventory` plugin needs to be enabled, so Ansible can use it. This is done in the `[inventory]` section in the `ansible.cfg` file.

```
[inventory]
enable_plugins   = ansilabnl.micetro.inventory, host_list, auto
cache            = yes
cache_plugin     = jsonfile
cache_prefix     = ansilabnl.micetro.inv
cache_timeout    = 3600
cache_connection = /tmp/ansilabnl.micetro.inventory_cache
```

With the following meaning:

- `cache`: Switch caching on and off
- `cache_plugin`: Which caching plugin to use
  - `jsonfile`
  - `yaml`
  - `pickle`
  - …
- `cache_prefix`: User defined prefix to use when creating the cache files
- `cache_connection`: Path in which the cache plugin will save the cache files
- `cache_timeout`: Timeout for the cache in seconds

Now the inventory plugin can be used with Ansible, like:

```
ansible-inventory -i /path/to/ansilabnl.micetro.inventory.yml --list
```

Or set the `ansilabnl.micetro.inventory.yml` as the Ansible inventory in the `ansible.cfg` file.

  

```
inventory = ansilabnl.micetro.inventory.yml
```

# 2.3. ansilabnl.micetro.ipinfo plugin

This Men&Mice IPInfo lookup plugin finds a lot of info about a specified IP address, defined in Men&Mice Micetro.

## 2.3.1. Options

- `ipaddress`: (required) The IP address that is examined
- `provider`: (required) Definition of the Men&Mice Micetro API provider.

## 2.3.2. Usage

The `ansilabnl.micetro.ipinfo` plugin delivers a complete set of information about an IP address, as it is delivered by the Men&Mice Micetro API.

Example usage:

*Listing 2. Get information on an IP address*

```
- name: Get all info for this IP address
  debug:
    var: ipinfo
  vars:
    ipinfo: "{{ query('ansilabnl.micetro.ipinfo', provider, '172.16.17.2') |
to_nice_json }}"
```

With output like (output shortened):

```
ok: [localhost] => {
    "ipinfo": {
        "addrRef": "IPAMRecords/11",
        "address": "172.16.17.2",
        "claimed": false,
        "customProperties": {
            "location": "At the attic"
        },
    }
}
```

  

# Chapter 3. Ansible modules

## 3.1. ansilabnl.micetro.user

Manage user accounts and user properties on Men&Mice Micetro

### 3.1.1. Options

- `authentication_type`: Authentication type to use. e.g. Internal, AD. Required if `state=present`.
- `descr`: Description of the user.
- `email`: The users email address.
- `groups`: Make the user a member of these groups.
- `name`: (required) Name of the user to create, remove or modify.
- `password`: Users password (plaintext). Required if `state=present`.
- `provider`: (required) Definition of the Men&Mice Micetro API provider.
- `roles`: Make the user a member of these roles.
- `state`: Should the users account exist or not. (`absent`, `present`)

### 3.1.2. Examples

*Listing 3. User example*

```
- name: Add the user 'johnd' as an admin
  ansilabnl.micetro.user:
    username: johnd
    password: password
    full_name: John Doe
    state: present
    authentication_type: internal
    roles:
        - Administrators (built-in)
        - DNS Administrators (built-in)
        - DHCP Administrators (built-in)
        - IPAM Administrators (built-in)
        - User Administrators (built-in)
        - Approvers (built-in)
        - Requesters (built-in)
    provider: "{{ provider }}"
  delegate_to: localhost

- name: Remove user 'johnd'
  ansilabnl.micetro.user:
    username: johnd
    state: absent
    provider: "{{ provider }}"
  delegate_to: localhost
```

## 3.2. ansilabnl.micetro.group

Manage groups on Men&Mice Micetro

### 3.2.1. Options

- `descr`: Description of the group.
- `name`: (required) Name of the group to create, remove or modify.
- `provider`: (required) Definition of the Men&Mice Micetro API provider.
- `roles`: List of roles to add to this group.
- `state`: Should the role exist or not. (`absent`, `present`)
- `users`: List of users to add to this group.

### 3.2.2. Examples

*Listing 4. Group example*

```
- name: Add the 'local' group
  ansilabnl.micetro.group:
    name: local
    desc: A local group
    state: present
    users:
      - johndoe
    roles:
      - IPAM Administrators (built-in)
  provider: "{{ provider }}"
  delegate_to: localhost

- name: Remove the 'local' group
  ansilabnl.micetro.group:
    name: local
    state: absent
    provider: "{{ provider }}"
  delegate_to: localhost
```

## 3.3. ansilabnl.micetro.role

Manage roles on Men&Mice Micetro

### 3.3.1. Options

- `descr`: Description of the role.
- `groups`: List of groups to add to this role
- `name`: (required) Name of the role to create, remove or modify.
- `provider`: (required) Definition of the Men&Mice Micetro API provider.
- `state`: Should the role exist or not. (`absent`, `present`)
- `users`: List of users to add to this role

### 3.3.2. Examples

  

*Listing 5. Role example*

```
- name: Add the 'local' role
  ansilabnl.micetro.role:
    name: local
    desc: A local role
    state: present
  provider: "{{ provider }}"
  delegate_to: localhost

- name: Remove the 'local' role
  ansilabnl.micetro.role:
    name: local
    state: absent
    provider: "{{ provider }}"
  delegate_to: localhost
```

# 3.4. ansilabnl.micetro.props

Manage custom properties in Men&Mice Micetro

## 3.4.1. Options

- `cloudtags`: Associated cloud tags.
- `defaultvalue`: Default value of the property.
- `dest`: (required) The section where to define the custom property.
- `listitems`: The items in the selection list.
- `mandatory`: Is the property mandatory.
- `multiline`: Is the property multiline.
- `name`: (required) Name of the property.
- `proptype`: Type of the property. These are not the types as described in the API, but the types as you can see them in the Men&Mice Management Console.
- `provider`: (required) Definition of the Men&Mice Micetro API provider.
- `readonly`: Is the property read only.
- `state`: The state of the properties or properties. (`absent`, `present`)
- `system`: Is the property system defined.
- `updateexisting`: Should objects be updated with the new values. Only valid when updating a property, otherwise ignored.

## 3.4.2. Examples

*Listing 6. Custom properties example*

```
- name: Set deinition for custom properties
  ansilabnl.micetro.props:
    name: location
    state: present
    proptype: text
    dest: zone
    provider: "{{ provider }}"
  delegate_to: localhost
```

# 3.5. ansilabnl.micetro.claimip

Claim IP addresses in DHCP in Men&Mice Micetro

## 3.5.1. Options

- `customproperties`: Custom properties for the IP address. These properties must already exist. See also [ansilabnl.micetro.props]
- `ipaddress`: (required) The IP address(es) to work on.
- `provider`: (required) Definition of the Men&Mice Micetro API provider.
- `state`: The state of the claim. (`absent`, `present`)

## 3.5.2. Examples

*Listing 7. Claim IP address example*

```
- name: Claim IP address
  ansilabnl.micetro.claimip:
    state: present
    ipaddress: 172.16.12.14
    provider: "{{ provider }}"
  delegate_to: localhost

- name: Release claim on IP addresses
  ansilabnl.micetro.claimip:
    state: present
    ipaddress:
      - 172.16.12.14
      - 172.16.12.15
      - 172.16.12.16
    provider: "{{ provider }}"
  delegate_to: localhost
```

# 3.6. ansilabnl.micetro.ipprops

Set properties on an IP address in Men&Mice Micetro

  

### 3.6.1. Options

- `deleteunspecified`: Clear properties that are not explicitly set.
- `ipaddress`: (required) The IP address(es) to work on.
- `properties`: (required) Custom properties for the IP address. These properties must already be defined.
- `provider`: (required) Definition of the Men&Mice Micetro API provider.
- `state`: Property present or not. (`absent`, `present`)

### 3.6.2. Examples

*Listing 8. IP address custom properties example*

```
- name: Set properties on IP
  ansilabnl.micetro.ipprops:
    state: present
    ipaddress: 172.16.12.14
    properties:
      claimed: false
      location: London
    provider: "{{ provider }}"
  delegate_to: localhost
```

## 3.7. ansilabnl.micetro.dhcp

Manage DHCP reservations on Men&Mice Micetro

### 3.7.1. Options

- `ddnshost`: The dynamic DNS host to place the entry in.
- `deleteunspecified`: Clear properties that are not explicitly set.
- `filename`: Filename to place the entry in.
- `ipaddress`: (required) The IP address(es) to make a reservation on. When the IP address is changed a new reservation is made. It is not allowed to make reservations in DHCP blocks.
- `macaddress`: (required) MAC address for the IP address.
- `name`: (required) Name of the reservation
- `nextserver`: Next server as DHCP option (bootp).
- `provider`: (required) Definition of the Men&Mice Micetro API provider.
- `servername`: Server to place the entry in.
- `state`: The state of the reservation. (`absent`, `present`)

### 3.7.2. Examples

  

*Listing 9. DHCP reservation example*

```
- name: Add a reservation for an IP address
  ansilabnl.micetro.dhcp:
    state: present
    name: myreservation
    ipaddress: 172.16.17.8
    macaddress: 44:55:66:77:88:99
    provider: "{{ provider }}"
  delegate_to: localhost
```

# 3.8. ansilabnl.micetro.zone

Manage DNS zones in Men&Mice Micetro

## 3.8.1. Options

- `adintegrated`: True if the zone is Active Directory integrated.
- `adpartition`: The AD partition if the zone is Active Directory integrated.
- `adreplicationtype`: Type of the AD replication.
- `authority`: Name of the DNS server that contains the zone or the string `[Active Directory]` if the zone is integrated in the Active Directory.
- `customproperties`: Custom properties for the zone. These properties must already exist. See also [ansilabnl.micetro.props].
- `dynamic`: Dynamic DNS zone.
- `masters`: The IP addresses of the master servers if the new zone is not a master zone.
- `name`: (required) Name of the zone.
- `nameserver`: Nameserver to define the zone on. Required if `state=present`.
- `provider`: (required) Definition of the Men&Mice Micetro API provider.
- `servtype`: Type of the master server.
- `state`: The state of the zone. (`absent`, `present`)

## 3.8.2. Examples

  

*Listing 10. Zone example*

```
- name: Create a new zone
  ansilabnl.micetro.zone:
    state: present
    name: example.com
    nameserver: ns1.example.com
    authority: mmsuite.example.net
    customproperties:
      location: Reykjavik
    provider: "{{ provider }}"
  delegate_to: localhost

- name: Release a zone
  ansilabnl.micetro.zone:
    state: absent
    name: example.com
    provider: "{{ provider }}"
  delegate_to: localhost
```

# 3.9. ansilabnl.micetro.dnsrecord

Manage DNS records in Men&Mice Micetro

In DNS it is very common to have multiple entries with the same name, as the example below shows.

*Listing 11. Multiple DNS entries for a single name*

```
mail01.example.net.   7200   IN   A      192.0.2.25
mail01.example.net.   7200   IN   A      192.0.2.143
mail01.example.net.   7200   IN   AAAA   2001:db8::25
mail01.example.net.   7200   IN   AAAA   2001:db8::587
```

To enable multiple records with the same name in the Ansible modules, there is no possibility to change a record, the only way is to add the new record with the updated data and remove the old one after that.

## 3.9.1. Options

- `aging`: The aging timestamp of dynamic records in AD integrated zones. Hours since January 1, 1601, UTC. Providing a non-zero value creates a dynamic record.
- `comment`: Comment string for the record. Note that only records in static DNS zones can have a comment string
- `data`: (required) The data that is added to the DNS record. The record data is a space-separated list, when the resource type is one of: `MX`, `SRV`, `NAPTR`, `CAA`, `CERT`, `HINFO` or `TLSA`.
  Example: `data: "100 10 U E2U+sip !^.*$!sip:customer-service@example.com! ."` For MX and SRV the hostname should be the short name and not the FQDN.
- `dnszone`: (required) The DNS zone where the action should take place.
- `enabled`: True if the record is enabled. If the record is disabled the value is false

  

- `name`: (required) The name of the DNS record. Can either be partially or fully qualified.
- `provider`: (required) Definition of the Men&Mice Micetro API provider.
- `rrtype`: Resource Record Type for this DNS record. Default is `A`.
- `state`: The state of the properties. (`absent`, `present`)
- `ttl`: The Time-To-Live of the DNS record.

### 3.9.2. Examples

*Listing 12. DNS record setting example*

```
- name: Set DNS record in zone for a defined name
  ansilabnl.micetro.dnsrecord:
    state: present
    name: beatles
    data: 172.16.17.2
    rrtype: A
    dnszone: example.net.
    provider: "{{ provider }}"
  delegate_to: localhost

- name: Set PTR record in zone for a defined name
  ansilabnl.micetro.dnsrecord:
    state: present
    name: "2.17.16.172.in-addr.arpa."
    data: beatles.example.net.
    rrtype: PTR
    dnszone: "17.16.172.in-addr.arpa."
    provider: "{{ provider }}"
  delegate_to: localhost

- name: Set MX record
  ansilabnl.micetro.dnsrecord:
    state: present
    name: beatles
    rrtype: MX
    dnszone: example.net.
    data: "10 ringo"
    ttl: 86400
    provider: "{{ provider }}"
  delegate_to: localhost
```

  

# Chapter 4. Example playbooks

To use Men&Mice Micetro Ansible Integration you need to create Ansible playbooks that utilize the functionality of Men&Mice Micetro.

Following are a couple of example playbooks for inspiration.

These playbooks have been tested extensively with different operating systems, versions of Ansible and Python. For a complete overview, have a look at the "`Testmatrix`" chapter.

Caveat: As the operating systems do not have all these combinations of Ansible and Python available, the tests where done in Python virtual environments.

All these playbooks are available in the `examples` directory.

## 4.1. play-user

*Listing 13. Add, delete or change a user*

```
---
#
# Add, delete and change users on Men&Mice Micetro example
#
# The file <ansible_topdir>/group_vars/all contains:
#
#    ---
#    provider:
#      mmurl: http://mmsuite.example.net
#      user: apiuser
#      password: apipasswd
#
- name: Men&Mice Micetro users test play
  hosts: localhost
  connection: local
  become: false

  tasks:
    - name: Get the free IP address and show info
      ansible.builtin.debug:
        msg:
          - "Ansible version    : {{ ansible_version.full }}"
          - "Python version     : {{ ansible_facts['python_version'] }}"
          - "Python executable  : {{ ansible_facts['python']['executable'] }}"

    - name: Add the user 'johnd' as an admin
      ansilabnl.micetro.user:
        username: johnd
        password: password
        full_name: John Doe
        state: present
        authentication_type: internal
        roles:
          - Administrators (built-in)
          - DNS Administrators (built-in)
```

```
        - DHCP Administrators (built-in)
        - IPAM Administrators (built-in)
        - User Administrators (built-in)
        - Approvers (built-in)
        - Requesters (built-in)
      provider: "{{ provider }}"

  - name: Check idempotency
    ansilabnl.micetro.user:
      username: johnd
      password: password
      full_name: John Doe
      state: present
      authentication_type: internal
      roles:
        - Administrators (built-in)
        - DNS Administrators (built-in)
        - DHCP Administrators (built-in)
        - IPAM Administrators (built-in)
        - User Administrators (built-in)
        - Approvers (built-in)
        - Requesters (built-in)
      provider: "{{ provider }}"

  - name: Change the groups
    ansilabnl.micetro.user:
      username: johnd
      password: password
      full_name: John Doe
      state: present
      authentication_type: internal
      roles:
        - Administrators (built-in)
        - User Administrators (built-in)
        - Approvers (built-in)
        - Requesters (built-in)
      provider: "{{ provider }}"

  - name: Check idempotency again
    ansilabnl.micetro.user:
      username: johnd
      password: password
      full_name: John Doe
      state: present
      authentication_type: internal
      roles:
        - Administrators (built-in)
        - User Administrators (built-in)
        - Approvers (built-in)
        - Requesters (built-in)
      provider: "{{ provider }}"

  - name: Remove the user again
    ansilabnl.micetro.user:
      username: johnd
      state: absent
      provider: "{{ provider }}"
```

## 4.2. play-group

*Listing 14. Add, delete or change a group*

```
---
#
# Add, delete and change groups on Men&Mice Micetro example
#
# The file <ansible_topdir>/group_vars/all contains:
#
#    ---
#    provider:
#      mmurl: http://mmsuite.example.net
#      user: apiuser
#      password: apipasswd
#
- name: Men&Mice Micetro users test play
  hosts: localhost
  connection: local
  become: false

  tasks:
    - name: Get the free IP address and show info
      ansible.builtin.debug:
        msg:
          - "Ansible version    : {{ ansible_version.full }}"
          - "Python version     : {{ ansible_facts['python_version'] }}"
          - "Python executable  : {{ ansible_facts['python']['executable'] }}"

    - name: Add the 'local' group
      ansilabnl.micetro.group:
        name: local
        desc: A local rgroup
        state: present
        users:
          - johndoe
          - angelina
        provider: "{{ provider }}"

    - name: Check idempotency
      ansilabnl.micetro.group:
        name: local
        desc: A local group
        state: present
        users:
          - johndoe
          - angelina
        provider: "{{ provider }}"

    - name: Add nonexisting user to group
      ansilabnl.micetro.group:
        name: local
        desc: A local group
        state: present
        users:
          - neverheardof
```

```
        provider: "{{ provider }}"
      ignore_errors: true                       # noqa: ignore-errors

    - name: Remove the 'local' group
      ansilabnl.micetro.group:
        name: local
        state: absent
        provider: "{{ provider }}"
```

## 4.3. play-role

*Listing 15. Add, delete or change a role*

```
---
#
# Add, delete and change roles on Men&Mice Micetro example
#
# The file <ansible_topdir>/group_vars/all contains:
#
#    ---
#    provider:
#      mmurl: http://mmsuite.example.net
#      user: apiuser
#      password: apipasswd
#
- name: Men&Mice Micetro users test play
  hosts: localhost
  connection: local
  become: false

  tasks:
    - name: Get the free IP address and show info
      ansible.builtin.debug:
        msg:
          - "Ansible version    : {{ ansible_version.full }}"
          - "Python version     : {{ ansible_facts['python_version'] }}"
          - "Python executable  : {{ ansible_facts['python']['executable'] }}"

    - name: Add the 'local' role
      ansilabnl.micetro.role:
        name: local
        desc: A local role
        state: present
        users:
          - johndoe
          - angelina
        provider: "{{ provider }}"

    - name: Check idempotency
      ansilabnl.micetro.role:
        name: local
        desc: A local role
        state: present
        users:
          - johndoe
```

```
          - angelina
        provider: "{{ provider }}"

    - name: Add nonexisting user to role
      ansilabnl.micetro.role:
        name: local
        desc: A local role
        state: present
        users:
          - neverheardof
        provider: "{{ provider }}"
      ignore_errors: true                       # noqa: ignore-errors

    - name: Remove the 'local' role
      ansilabnl.micetro.role:
        name: local
        state: absent
        provider: "{{ provider }}"
```

## 4.4. play-props

*Listing 16. Add, delete or change custom properties on assets*

```
---
#
# Set, delete and change custom properties on Men&Mice Micetro example
#
# The file <ansible_topdir>/group_vars/all contains:
#
#    ---
#    provider:
#      mmurl: http://mmsuite.example.net
#      user: apiuser
#      password: apipasswd
#
- name: Men&Mice Micetro Custom Properties test play
  hosts: localhost
  connection: local
  become: false

  tasks:
    - name: Ansible information
      ansible.builtin.debug:
        msg:
          - "Ansible version    : {{ ansible_version.full }}"
          - "Python version     : {{ ansible_facts['python_version'] }}"
          - "Python executable : {{ ansible_facts['python']['executable'] }}"

    - name: Set text property
      ansilabnl.micetro.props:
        state: present
        name: MyProperty
        proptype: text
        dest: dnsserver
        listitems:
```

```
        - John
        - Paul
        - Ringo
        - George
      provider: "{{ provider }}"
    delegate_to: localhost

  - name: Check idempotentie
    ansilabnl.micetro.props:
      state: present
      name: MyProperty
      proptype: text
      dest: dnsserver
      listitems:
        - John
        - Paul
        - Ringo
        - George
      provider: "{{ provider }}"
    delegate_to: localhost

  - name: Change type - not allowed
    ansilabnl.micetro.props:
      state: present
      name: MyProperty
      proptype: yesno
      dest: dnsserver
      listitems:
        - John
        - Paul
        - Ringo
        - George
      provider: "{{ provider }}"
    delegate_to: localhost

  - name: Change list around
    ansilabnl.micetro.props:
      state: present
      name: MyProperty
      proptype: text
      dest: dnsserver
      listitems:
        - George
        - John
        - Paul
        - Ringo
      provider: "{{ provider }}"
    delegate_to: localhost

  - name: Remove property
    ansilabnl.micetro.props:
      state: absent
      name: MyProperty
      proptype: text
      dest: dnsserver
      provider: "{{ provider }}"
    delegate_to: localhost
```

  

```
    - name: Remove property - again
      ansilabnl.micetro.props:
        state: absent
        name: MyProperty
        proptype: yesno
        dest: dnsserver
        provider: "{{ provider }}"
      delegate_to: localhost
```

## 4.5. play-claimip

*Listing 17. Claim IP addresses in one or more ranges*

```
---
#
# Claim and release an IP address on Men&Mice Micetro example
#
# The file <ansible_topdir>/group_vars/all contains:
#
#    ---
#    provider:
#      mmurl: http://mmsuite.example.net
#      user: apiuser
#      password: apipasswd
#
#
- name: Men&Mice Micetro ClaimIP test play
  hosts: localhost
  connection: local
  become: false

  tasks:
    - name: Ansible information
      ansible.builtin.debug:
        msg:
          - "Ansible version   : {{ ansible_version.full }}"
          - "Python version    : {{ ansible_facts['python_version'] }}"
          - "Python executable : {{ ansible_facts['python']['executable'] }}"

    - name: Claim IP address
      ansilabnl.micetro.claimip:
        state: present
        ipaddress: 172.16.12.14
        provider: "{{ provider }}"

    - name: Check idempotentie
      ansilabnl.micetro.claimip:
        state: present
        ipaddress: 172.16.12.14
        provider: "{{ provider }}"

    - name: Unclaim IP address
      ansilabnl.micetro.claimip:
        state: present
        ipaddress: 172.16.12.14
        provider: "{{ provider }}"

    # This task claims an IP address that cannot exit
    # and returns a warning because of that
    - name: Claim erroneous IP address
      ansilabnl.micetro.claimip:
        state: present
        ipaddress: 456.978.12.14
        provider: "{{ provider }}"
```

## 4.6. play-dhcp

*Listing 18. Make and release DHCP reservations*

```
---
#
# Make a DHCP reservation and release it on Men&Mice Micetro example
#
# The file <ansible_topdir>/group_vars/all contains:
#
#    ---
#    provider:
#      mmurl: http://mmsuite.example.net
#      user: apiuser
#      password: apipasswd
#
- name: Men&Mice Micetro DHCP test play
  hosts: localhost
  connection: local
  become: false

  tasks:
    - name: Ansible information
      ansible.builtin.debug:
        msg:
          - "Ansible version    : {{ ansible_version.full }}"
          - "Python version     : {{ ansible_facts['python_version'] }}"
          - "Python executable : {{ ansible_facts['python']['executable'] }}"

    - name: Add a reservation for an IP address
      ansilabnl.micetro.dhcp:
        state: present
        name: myreservation
        ipaddress: 172.16.17.8
        macaddress: 44:55:66:77:88:00
        provider: "{{ provider }}"
      delegate_to: localhost

    - name: check idempotentie
      ansilabnl.micetro.dhcp:
        state: present
        name: myreservation
        ipaddress: 172.16.17.8
        macaddress: 44:55:66:77:88:00
        provider: "{{ provider }}"
      delegate_to: localhost

    # Changing the MAC address of a reservation is not allowed, as this
    # would alter the reservation. To achieve this, release the reservation
    # and reclaim it.
    - name: change mac
      ansilabnl.micetro.dhcp:
        state: present
        name: myreservation
        ipaddress: 172.16.17.8
        macaddress: 44:55:66:77:88:99
```

```
      provider: "{{ provider }}"
    delegate_to: localhost

  - name: change ip
    ansilabnl.micetro.dhcp:
      state: present
      name: myreservation
      ipaddress: 172.16.17.9
      macaddress: 44:55:66:77:88:99
      provider: "{{ provider }}"
    delegate_to: localhost

  - name: change name
    ansilabnl.micetro.dhcp:
      state: present
      name: movemyreservation
      ipaddress: 172.16.17.9
      macaddress: 44:55:66:77:88:99
      provider: "{{ provider }}"
    delegate_to: localhost

  - name: delete reservation (wrong one)
    ansilabnl.micetro.dhcp:
      state: absent
      name: movemyreservation
      ipaddress: 172.16.17.9
      macaddress: 44:55:66:77:88:99
      provider: "{{ provider }}"
    delegate_to: localhost

  - name: delete reservation (correct one)
    ansilabnl.micetro.dhcp:
      state: absent
      name: myreservation
      ipaddress: 172.16.17.8
      macaddress: 44:55:66:77:88:99
      provider: "{{ provider }}"
    delegate_to: localhost

  - name: create reservation in invalid range
    ansilabnl.micetro.dhcp:
      state: present
      name: reservationnonet
      ipaddress: 172.16.17.58
      macaddress: 44:55:66:77:88:99
      provider: "{{ provider }}"
    delegate_to: localhost
```

## 4.7. play-zone

  

*Listing 19. Add, delete or change a DNS zone*

```
---
#
# The file <ansible_topdir>/group_vars/all contains:
#
#    ---
#    provider:
#      mmurl: http://mmsuite.example.net
#      user: apiuser
#      password: apipasswd
#
- name: Men&Mice Micetro zone test play
  hosts: localhost
  connection: local
  become: false

  tasks:
    - name: Ansible information
      ansible.builtin.debug:
        msg:
          - "Ansible version   : {{ ansible_version.full }}"
          - "Python version    : {{ ansible_facts['python_version'] }}"
          - "Python executable : {{ ansible_facts['python']['executable'] }}"

    - name: Ensure the zone
      ansilabnl.micetro.zone:
        state: present
        name: example.com
        nameserver: mandm.example.com
        authority: mandm.example.net
        masters: mandm.example.net
        servtype: master
        customproperties:
          owner: Me, myself and I
          place: Netherlands
        provider: "{{ provider }}"
      delegate_to: localhost

    - name: Remove the zone
      ansilabnl.micetro.zone:
        state: absent
        name: example.com
        provider: "{{ provider }}"
      delegate_to: localhost
```

# 4.8. play-dnsrecord

*Listing 20. Add and change a DNS record*

```
---
#
# Set and change a DNS record on Men&Mice Micetro example
#
# The file <ansible_topdir>/group_vars/all contains:
```

```yaml
#
#    ---
#    provider:
#      mmurl: http://mmsuite.example.net
#      user: apiuser
#      password: apipasswd
#
- name: Men&Mice Micetro DNSRecord test play
  hosts: localhost
  connection: local
  become: false

  tasks:
    - name: Ansible information
      ansible.builtin.debug:
        msg:
          - "Ansible version   : {{ ansible_version.full }}"
          - "Python version    : {{ ansible_facts['python_version'] }}"
          - "Python executable : {{ ansible_facts['python']['executable'] }}"

    - name: Set DNS record
      ansilabnl.micetro.dnsrecord:
        state: present
        name: beatles
        rrtype: A
        dnszone: testzone
        data: 192.168.10.12
        comment: From The API side
        ttl: 86400
        provider: "{{ provider }}"
      delegate_to: localhost

    - name: Check idempotentie
      ansilabnl.micetro.dnsrecord:
        state: present
        name: beatles
        rrtype: A
        dnszone: testzone
        data: 192.168.10.12
        comment: From The API side
        ttl: 86400
        provider: "{{ provider }}"
      delegate_to: localhost

    - name: Set DNS record with erroneous values
      ansilabnl.micetro.dnsrecord:
        state: present
        name: beatles
        rrtype: AAAA
        dnszone: testzone
        data: 192.168.10.127
        comment: From The API side
        ttl: apple
        provider: "{{ provider }}"
      delegate_to: localhost
      ignore_errors: true                       # noqa: ignore-errors
```

  

```
    - name: Change record
      ansilabnl.micetro.dnsrecord:
        state: present
        name: beatles
        rrtype: A
        dnszone: testzone
        data: 192.168.10.14
        comment: From The API side
        provider: "{{ provider }}"
      delegate_to: localhost

    - name: Add records to non existing zone
      ansilabnl.micetro.dnsrecord:
        state: present
        name: beatles
        rrtype: A
        dnszone: notthetestzone
        data: 192.168.90.14
        comment: Welcome to the error
        provider: "{{ provider }}"
      delegate_to: localhost
      ignore_errors: true                       # noqa: ignore-errors

    - name: Use a very invalid IP address
      ansilabnl.micetro.dnsrecord:
        state: present
        name: beatles
        rrtype: A
        dnszone: testzone
        data: 192.168.390.14
        comment: Welcome to the error
        provider: "{{ provider }}"
      delegate_to: localhost
      ignore_errors: true                       # noqa: ignore-errors

    - name: Remove record
      ansilabnl.micetro.dnsrecord:
        state: absent
        name: beatles
        dnszone: notthetestzone
        data: 192.168.90.14
        provider: "{{ provider }}"
      delegate_to: localhost

    - name: Remove record - again
      ansilabnl.micetro.dnsrecord:
        state: absent
        name: beatles
        dnszone: notthetestzone
        data: 192.168.90.14
        provider: "{{ provider }}"
      delegate_to: localhost
```

  

# 4.9. play-freeip

*Listing 21. Find free IP addresses in a range or ranges*

```
---
#
# Find a set of free IP addresses in a range on Men&Mice Micetro example
#
# The file <ansible_topdir>/group_vars/all contains:
#
#    ---
#    provider:
#      mmurl: http://mmsuite.example.net
#      user: apiuser
#      password: apipasswd
#
- name: Men&Mice Micetro FreeIP test play
  hosts: localhost
  connection: local
  become: false

  vars:
    network:
      - examplenet

  tasks:
    - name: Set free IP addresses as a fact
      ansible.builtin.set_fact:
        freeips: "{{ query('ansilabnl.micetro.freeip',
                           provider,
                           network,
                           multi=25,
                           claim=60,
                           excludedhcp=True,
                           ping=True)
                }}"

    - name: Get the free IP address and show info
      ansible.builtin.debug:
        msg:
          - "Free IPs           : {{ freeips }}"
          - "Queried network(s) : {{ network }}"
          - "Ansible version    : {{ ansible_version.full }}"
          - "Python version     : {{ ansible_facts['python_version'] }}"
          - "Python executable  : {{ ansible_facts['python']['executable'] }}"

    - name: Loop over IP addresses
      ansible.builtin.debug:
        msg:
          - "Next free IP       : {{ item }}"
      loop: "{{ freeips }}"
```

  

## 4.10. play-ipinfo

*Listing 22. Collect a lot of info concerning an IP address*

```
---
#
# Get all info for an IP address on Men&Mice Micetro example
#
# The file <ansible_topdir>/group_vars/all contains:
#
#    ---
#    provider:
#      mmurl: http://mmsuite.example.net
#      user: apiuser
#      password: apipasswd
#
- name: Men&Mice Micetro IP Info test play
  hosts: localhost
  connection: local
  become: false

  tasks:
    - name: Get get IP info
      ansible.builtin.set_fact:
        ipinfo: "{{ query('ansilabnl.micetro.ipinfo', provider, '172.16.17.2')
| to_nice_json }}"

    - name: Show Ansible and Python information
      ansible.builtin.debug:
        msg:
          - "Ansible version    : {{ ansible_version.full }}"
          - "Python version     : {{ ansible_facts['python_version'] }}"
          - "Python executable  : {{ ansible_facts['python']['executable'] }}"

    - name: Show all infor for this IP address
      ansible.builtin.debug:
        var: ipinfo

    # This task tries to get the information for a non-existing IP address
    # which results in a fatal `Object not found for reference` error
    - name: Get get IP info for a non existing IP address
      ansible.builtin.set_fact:
        ipinfo: "{{ query('ansilabnl.micetro.ipinfo', provider, '390.916.17.2')
| to_nice_json }}"
      ignore_errors: true                       # noqa: ignore-errors
```

## 4.11. play_it_all

*Listing 23. Example of a playbook that combines functionality*

```
---
- name: Men&Mice Micetro test play
  hosts: localhost
  connection: local
```

```
  become: false

  vars:
    network: examplenet

  tasks:
    # Some extra information about Ansible and the used
    # Python version
    - name: Ansible information
      ansible.builtin.debug:
        msg:
          - "Ansible version   : {{ ansible_version.full }}"
          - "Python version    : {{ ansible_facts['python_version'] }}"
          - "Python executable : {{ ansible_facts['python']['executable'] }}"

    # The `ipaddr` filter needs the Python `netaddr` module, so make sure
    # this is installed
    # The `ipaddr` is used to determine the reverse IP address
    #
    # For example:
    #   vars:
    #     ipa4: "172.16.17.2"
    #     ipa6: "2001:785:beef:1:f2c4:8f9d:b554:e614"
    #
    #   - "Forward IPv4 address : {{ ipa4 }}"
    #   - "Forward IPv4 address : {{ ipa4 }}"
    #   - "Reverse IPv4 address : {{ ipa4 | ipaddr('revdns') }}"
    #   - "Reverse IPv6 address : {{ ipa6 | ipaddr('revdns') }}"
    #   - "Reverse IPv4 zone    : {{ (ipa4 | ipaddr('revdns')).split('.')[1:]
| join('.') }}"
    #   - "Reverse IPv6 zone    : {{ (ipa6 | ipaddr('revdns')).split('.')[16:]
| join('.') }}"
    #
    # The reverse zones are split on '.' and only the last part is
    # used (in this example). The reverse for IPv4 assumes a '/24' network
    # and the '16' in the IPv6 zone conversion is for a '/64' network. Adapt
these to your
    # own needs (e.g. '2' for a '/16' network on IPv4 or '20' for an IPv6 '/48'
net.

    - name: Ensure the netaddr module is installed for Python 2
      ansible.builtin.pip:
        name: netaddr
        state: present
      when: ansible_facts['python_version'] is version('3', '<')
      become: true

    - name: Ensure the netaddr module is installed for Python 3
      ansible.builtin.pip:
        name: netaddr
        state: present
        executable: pip3
      when: ansible_facts['python_version'] is version('3', '>=')
      become: true

    - name: define custom properties for IP addresses
      ansilabnl.micetro.props:
```

```
        name: location
        state: present
        proptype: text
        dest: ipaddress
        provider: "{{ provider }}"

    # The above example defines just a single property.
    # Defining multiple properties can be achieved by using
    # the Ansible loop functionality.
    #
    # - name: Example of multiple properties
    #   ansilabnl.micetro.props:
    #      name: "{{ item.name }}"
    #      state: "{{ item.state }}"
    #      proptype: "{{ item.proptype }}"
    #      dest: "{{ item.dest }}"
    #  loop:
    #    - name: location
    #      state: present
    #      proptype: text
    #      dest: ipaddress
    #    - name: owner
    #      state: present
    #      proptype: text
    #      dest: ipaddress

    # When running an Ansible lookup plugin, this lookup action takes
    # place every time the variable is referenced. So it will not be
    # possible to claim an IP address for further reference, this way.
    # This has to do with the way Ansible works.  A solution for this
    # is to assign all collected free IP addresses to an Ansible fact,
    # but here you need to make sure the factname is not used over
    # multiple hosts.
    - name: get free IP addresses and set it as a fact
      ansible.builtin.set_fact:
        freeips: "{{ query('ansilabnl.micetro.freeip', provider, network,
claim=60, excludedhcp=True) }}"

    - name: Get the free IP address and show info
      ansible.builtin.debug:
        msg:
          - "Free IPs           : {{ freeips }}"
          - "Queried network(s) : {{ network }}"

    # Make a DHCP reservation for this address
    # So claim it after DNS setting.
    - name: Reservation on IP address
      ansilabnl.micetro.dhcp:
        state: present
        name: testhost
        ipaddress: "{{ freeips }}"
        macaddress: "de:ad:be:ef:16:10"
        provider: "{{ provider }}"
      delegate_to: localhost

    - name: Set properties on IP
      ansilabnl.micetro.ipprops:
```

```
        state: present
        ipaddress: "{{ freeips }}"
        properties:
          claimed: false
          location: London
        provider: "{{ provider }}"
      delegate_to: localhost

    - name: Ensure the zone
      ansilabnl.micetro.zone:
        state: present
        name: thetestzone.com
        nameserver: mandm.example.com
        authority: mandm.example.net
        masters: mandm.example.net
        servtype: master
        provider: "{{ provider }}"
      delegate_to: localhost

    # The `ansilabnl.micetro.freeip` plugin always returns a list, but the
request was for just 1
    # IP address. The `ansilabnl.micetro.dnsrecord` only needs a single IP
address. That's why the
    # list-slice `[0]` is used.
    - name: Set a DNS record for the claimed IP
      ansilabnl.micetro.dnsrecord:
        dnszone: testzone
        name: testhost
        data: "{{ freeips[0] }}"
        provider: "{{ provider }}"
      delegate_to: localhost

    - name: Set a PTR DNS record for the claimed IP
      ansilabnl.micetro.dnsrecord:
        dnszone: "{{ (freeips[0] |
ansible.utils.ipaddr('revdns')).split('.')[1:]  | join('.') }}"
        name: "{{ freeips[0] | ansible.utils.ipaddr('revdns') }}"
        data: "testhost.testzone."
        rrtype: PTR
        provider: "{{ provider }}"
      delegate_to: localhost

    # The `ansilabnl.micetro.ipinfo` returns all known information of an IP
    # address. This can be used to query certain properties, or
    # for debugging.
    - name: Get all info for this IP address
      ansible.builtin.debug:
        var: freeipinfo
      vars:
        freeipinfo: "{{ query('ansilabnl.micetro.ipinfo', provider, freeips[0])
| to_nice_json }}"

    - name: Renew properties on IP
      ansilabnl.micetro.ipprops:
        state: present
        ipaddress: "{{ freeips }}"
        properties:
```

```
          claimed: false
          location: Madrid
        provider: "{{ provider }}"
      delegate_to: localhost

    - name: Get all info for this IP address
      ansible.builtin.debug:
        var: freeipinfo
      vars:
        freeipinfo: "{{ query('ansilabnl.micetro.ipinfo', provider, freeips[0])
| to_nice_json }}"

    - name: Remove properties of IP
      ansilabnl.micetro.ipprops:
        state: present
        ipaddress: "{{ freeips }}"
        deleteunspecified: true
        properties:
          claimed: false
        provider: "{{ provider }}"
      delegate_to: localhost

    - name: Get all info for this IP address
      ansible.builtin.debug:
        var: freeipinfo
      vars:
        freeipinfo: "{{ query('ansilabnl.micetro.ipinfo', provider, freeips[0])
| to_nice_json }}"

    - name: Remove reservation on IP address
      ansilabnl.micetro.dhcp:
        state: absent
        name: testhost
        ipaddress: "{{ freeips }}"
        macaddress: "de:ad:be:ef:16:10"
        provider: "{{ provider }}"
      delegate_to: localhost

    - name: Get all info for this IP address
      ansible.builtin.debug:
        var: freeipinfo
      vars:
        freeipinfo: "{{ query('ansilabnl.micetro.ipinfo', provider, freeips[0])
| to_nice_json }}"

    - name: Remove DNS record for the claimed IP
      ansilabnl.micetro.dnsrecord:
        state: absent
        dnszone: testzone
        name: testhost
        data: "{{ freeips[0] }}"
        provider: "{{ provider }}"
      delegate_to: localhost

    - name: Remove the PTR DNS record for the claimed IP
      ansilabnl.micetro.dnsrecord:
        state: absent
```

```
        dnszone: "{{ (freeips[0] |
ansible.utils.ipaddr('revdns')).split('.')[1:]  | join('.') }}"
        name: "{{ freeips[0] | ansible.utils.ipaddr('revdns') }}"
        data: "testhost.testzone."
        rrtype: PTR
        provider: "{{ provider }}"
      delegate_to: localhost

    - name: Get all info for this IP address
      ansible.builtin.debug:
        var: freeipinfo
      vars:
        freeipinfo: "{{ query('ansilabnl.micetro.ipinfo', provider, freeips[0])
| to_nice_json }}"

    - name: Ensure the zone absent
      ansilabnl.micetro.zone:
        state: absent
        name: thetestzone.com
        nameserver: mandm.example.com
        authority: mandm.example.net
        masters: mandm.example.net
        servtype: master
        provider: "{{ provider }}"
      delegate_to: localhost
```

# Chapter 5. Credential matrix

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| ansilabnl.micetro.claimip.py | | | | * | | | |
| ansilabnl.micetro.dhcp | | | * | * | | | |
| ansilabnl.micetro.dnsrecord | | * | | | | | |
| ansilabnl.micetro.group | | | | | * | | |
| ansilabnl.micetro.ipprops | | | * | | | | |
| ansilabnl.micetro.props | * | * | * | * | * | | |
| ansilabnl.micetro.role | | | | | * | | |
| ansilabnl.micetro.user | | | | | * | | |
| ansilabnl.micetro.zone | | * | | | | | |
| ansilabnl.micetro.inventory | | | | * | | | |
| ansilabnl.micetro.freeip | | | | * | | | |
| ansilabnl.micetro.ipinfo | | | | * | | | |

*Table 1. Module and plugin credentials needed*

1. Administrators (built-in)
2. DNS Administrators (built-in)
3. DHCP Administrators (built-in)
4. IPAM Administrators (built-in)
5. User Administrators (built-in)
6. Approvers (built-in)
7. Requesters (built-in)

## 5.1. Remarks

- The `ansilabnl.micetro.props` module manages custom properties for all types, like DNS servers, DHCP servers, zones, IP ranges etc. When using the module for a type when no modify rights are granted, an error will occur. It is possible to grant less rights and allow only to modify a subset of the record types.

  

# Chapter 6. Testmatrix

Below is an overview of the conducted tests for the Ansible modules and the plugins.

After a fresh install, all systems first had a complete update (`yum -y update` for Red Hat based machines, `apt-get update; apt-get dist-upgrade` for Debian based machines and `freebsd-update fetch; freebsd-update install` for FreeBSD) followed by a reboot.

When ansible, Python2, Python3 and virtualenv where not installed, yet, these packages where installed first, to make sure all tests can be run.

Some systems (like CentOS6 and Ubuntu 18.04) have a default Ansible version below 2.7. This immediately reflects in the every test failing. On these systems Ansible is installed through Python PIP, to ensure a valid Ansible version. If possible an Ansible version for Python3 was chosen. At the time of writing (2022-05-30) the latest stable version of Ansible is 2.9.9.

For CentOS6 the CentOS Software Collection for Python 2 and 3 was installed, as both original packages are too old for Ansible 2.7+ (From `centos-release-scl`) CentOS6 is still maintained, but not all the different combinations where tested, as not all requirements where met.

Running on Ubuntu 16 with Ansible 2.7 and Python 2.7 results in a `No module named errors`. This is a known error and it was fixed in 2.8.

| | Python 2 | | | Python 3 | | | Native | |
|---|---|---|---|---|---|---|---|---|
| **Ansible version** | **2.7** | **2.8** | **2.9** | **2.7** | **2.8** | **2.9** | **Ansible** | **Python** |
| CentOS6 | ✖ | ✖ | ✔ | ✖ | ✖ | ✔ | 2.9.9 | 3.6.9 |
| CentOS7 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | 2.9.9 | 2.7.5 |
| CentOS8 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | 2.9.9 | 3.6.8 |
| RHEL 8.2 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | 2.9.9 | 3.6.8 |
| Ubuntu 16.04 | ✖ | ✔ | ✔ | ✔ | ✔ | ✔ | 2.9.9 | 3.5.2 |
| Ubuntu 18.04 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | 2.9.9 | 2.7.17 |
| Ubuntu 20.04 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | 2.9.6 | 3.8.2 |
| Debian 10 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | 2.7.2 | 3.7.3 |
| FreeBSD 12 | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | 2.8.11 | 3.7.7 |

*Table 2. Ansible, Python and OS testmatrix*