Setting the remote environment

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\[ansible-devel][docs][docsite][rst][user_guide]playbooks_environment.rst, line 6)

Unknown directive type "versionadded".

```
.. versionadded:: 1.1
```

You can use the `environment` keyword at the play, block, or task level to set an environment variable for an action on a remote host. With this keyword, you can enable using a proxy for a task that does http requests, set the required environment variables for language-specific version managers, and more.

When you set a value with `environment:` at the play or block level, it is available only to tasks within the play or block that are executed by the same user. The `environment:` keyword does not affect Ansible itself, Ansible configuration settings, the environment for other users, or the execution of other plugins like lookups and filters. Variables set with `environment:` do not automatically become Ansible facts, even when you set them at the play level. You must include an explicit `gather_facts` task in your playbook and set the `environment` keyword on that task to turn these values into Ansible facts.

Setting the remote environment in a task

You can set the environment directly at the task level.

```
- hosts: all
  remote_user: root

  tasks:

    - name: Install cobbler
      ansible.builtin.package:
        name: cobbler
        state: present
      environment:
        http_proxy: http://proxy.example.com:8080
```

You can re-use environment settings by defining them as variables in your play and accessing them in a task as you would access any stored Ansible variable.

```
- hosts: all
  remote_user: root

  # create a variable named "proxy_env" that is a dictionary
  vars:
    proxy_env:
      http_proxy: http://proxy.example.com:8080

  tasks:

    - name: Install cobbler
      ansible.builtin.package:
        name: cobbler
        state: present
      environment: "{{ proxy_env }}"
```

You can store environment settings for re-use in multiple playbooks by defining them in a group_vars file.

```
---
# file: group_vars/boston

ntp_server: ntp.bos.example.com
backup: bak.bos.example.com
proxy_env:
  http_proxy: http://proxy.bos.example.com:8080
  https_proxy: http://proxy.bos.example.com:8080
```

You can set the remote environment at the play level.

```
- hosts: testing

  roles:
     - php
     - nginx

  environment:
    http_proxy: http://proxy.example.com:8080
```

These examples show proxy settings, but you can provide any number of settings this way.

Working with language-specific version managers

Some language-specific version managers (such as rbenv and nvm) require you to set environment variables while these tools are in use. When using these tools manually, you usually source some environment variables from a script or from lines added to your shell configuration file. In Ansible, you can do this with the environment keyword at the play level.

```
---
### A playbook demonstrating a common npm workflow:
# - Check for package.json in the application directory
# - If package.json exists:
#   * Run npm prune
#   * Run npm install

- hosts: application
  become: false

  vars:
    node_app_dir: /var/local/my_node_app

  environment:
    NVM_DIR: /var/local/nvm
    PATH: /var/local/nvm/versions/node/v4.2.1/bin:{{ ansible_env.PATH }}

  tasks:
  - name: Check for package.json
    ansible.builtin.stat:
      path: '{{ node_app_dir }}/package.json'
    register: packagejson

  - name: Run npm prune
    ansible.builtin.command: npm prune
    args:
      chdir: '{{ node_app_dir }}'
    when: packagejson.stat.exists

  - name: Run npm install
    community.general.npm:
      path: '{{ node_app_dir }}'
    when: packagejson.stat.exists
```

Note

The example above uses `ansible_env` as part of the PATH. Basing variables on `ansible_env` is risky. Ansible populates `ansible_env` values by gathering facts, so the value of the variables depends on the remote_user or become_user Ansible used when gathering those facts. If you change remote_user/become_user the values in `ansible_env` may not be the ones you expect.

Warning

Environment variables are normally passed in clear text (shell plugin dependent) so they are not a recommended way of passing secrets to the module being executed.

You can also specify the environment at the task level.

```
---
- name: Install ruby 2.3.1
  ansible.builtin.command: rbenv install {{ rbenv_ruby_version }}
  args:
    creates: '{{ rbenv_root }}/versions/{{ rbenv_ruby_version }}/bin/ruby'
  vars:
    rbenv_root: /usr/local/rbenv
    rbenv_ruby_version: 2.3.1
  environment:
    CONFIGURE_OPTS: '--disable-install-doc'
    RBENV_ROOT: '{{ rbenv_root }}'
    PATH: '{{ rbenv_root }}/bin:{{ rbenv_root }}/shims:{{ rbenv_plugins }}/ruby-build/bin:{{ ansible_env.PAT
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\[ansible-devel][docs][docsite][rst][user_guide]playbooks_environment.rst, line 146)

Unknown directive type "seealso".

```
   .. seealso::

      :ref:`playbooks_intro`
          An introduction to playbooks
      `User Mailing List <https://groups.google.com/group/ansible-devel>`_
          Have a question?  Stop by the google group!
      :ref:`communication_irc`
          How to join Ansible chat channels
```