**Гайд по поднятию WSL**

- активировал режим разработчика + компонент винды "Linux subsystem"

- выкачал Ubuntu дистрибутив из windows store

- запустил и назначил убунту креды:

username: egor

password: as for win user

- запустил bash и выполнил:

--- sudo apt-get update

--- sudo apt-get -y install python-pip python-dev libffi-dev libssl-dev - установка Pip

**Команды Баша**:

- cd /mnt/c/users/... - переход по папкам

- touch file.yml - создание плейбуки в текущей папке

- ansible-playbook test.yml --connection=local - запуск плейбуки

**Требования Ансибла 2.3.0.0**:

для хостов: SSH и Python 2.5 (or Python 2.4 with the Python simplejson library)

control-machine: Python 2.6 or later

**PLAYBOOK**

Структура

Playbook.yml

---

# This is a YAML comment

hosts:

strategy:

become: true

vars:

somevar: testvalue

tasks:

-name: test-name

apt: name=nginx

notify: restart service

-name: second-task

user: name=deploy group=web

register: somevar

ignore\_errors: True

-name: configure...

meta: flush\_handlers

handlers:

-name: restart service

service: serv-name

--- it's script beginning

Playbook: скрипт исполняемый ансиблом (состоит из плеев)

Структура одного плэя (play):

hosts: хосты (физические компы) для которых выполняется скрипт

strategy: linear – переход к следующей таске выполняется только после выполнения предыдущей всеми хостами; free – переход к след. таске осуществляется в рамках выполнениях предыдущей таски одним хостом.

vars: переменные объявленные в рамках плейбуки

var\_files: файлы содержащие переменные

ignore\_errors: указывает на то, можно ли продолжить дальнейшее выполнение тасков если произошла ошибка в текущей

register: запись результата выполнения модуля в переменную

become: позволяет стать другим юзером

notify: указывается в таске и принимает имя хендлера (или имя события, который слушает хендлер listen:), который должен отработать если таск изменил состояние сервера;

listen: часть хендлера, принимает имя события, которые нотифицирует таска (вся эта хрень нужна для нотифаинга множества хендлеров за раз в рамках плея)

local\_action: указывается в таске, который принимает модуль, который должен выполниться на ансибловской машине (т.е. текущей);

run\_once: таска в плее исполнится только один раз (полезно для локальных тасков);

delegate\_to: указывается в таске, который принимает хост, на котором должна выполниться таска;

serial: указывает число параллельных исполнений плея для хостов, либо процент от количества хостов, или вообще коллекцию разных значений, которые будут обрабатываться по порядку (первая итерация с 1-ым количеством хостов, вторая со вторым и т.д.);

max\_fail\_percentage: пороговый процент зафейленных хостов, превысив который плей будет считаться зафейленным;

handlers: тоже что и таск, но отрабатывает только в случае вызова notify из таска (хэндлер отрабатывает только один раз (отдельно для tasks, pre\_tasks, post\_tasks) и только после выполнения всех тасков Плэя, хэндлеры отрабатывают в порядке их определения в Плэе).

meta: flush\_handlers принудительно исполняет хендлеры

tasks: действия (связанные с модулем), исполняемые для хостов по правилам

--- таски выполняются параллельно для всех хостов

--- следующая таска не начнется пока не выполнится предыдущая для всех хостов

--- таски выполняются в заданном порядке

--- apt, user и др. - ансибл модули, позволяющие делать определенные действия

Правила Yaml и другие при написании плейбуки

\*\* Booleans: 'yes' used for modules' parameters and 'true' for other places in playbook.

\*\* Arrays: [My Fair Lady, Oklahoma, "The Pirates of Penzance"]

\*\* Dictionaries: {address: "742 Evergreen Terrace", city: Springfield, state: North Takoma}

\*\* Line Folding (для разбиения большого числа параметров на несколько строк):



**Ансибл конфиг файл (ansible.cfg)**

может задавать различные переменные используемые инвентори файлом (для задания дефолтового юзера и др.),

кастомные переменные используемые в плейбуках и т.п.

Параметры инвентори файла, дефолтовые значения которых могут быть переопределены:

| **Behavioral inventory parameter** | **ansible.cfg option** |
| --- | --- |
| ansible\_port | remote\_port |
| ansible\_user | remote\_user |
| ansible\_private\_key\_file | private\_key\_file |
| ansible\_shell\_type | executable (see the following paragraph) |

Встроенные параметры, определяемые ансиблом и доступные в плейбуке

| **Parameter** | **Description** |
| --- | --- |
| hostvars | A dict whose keys are Ansible hostnames and values are dicts that map variable names to values |
| inventory\_hostname | Fully qualified domain name of the current host as known by Ansible (e.g., myhost.example.com) |
| inventory\_hostname\_short | Name of the current host as known by Ansible, without the domain name (e.g., myhost) |
| group\_names | A list of all groups that the current host is a member of |
| groups | A dict whose keys are Ansible group names and values are a list of hostnames that are members of the group. Includes all and ungrouped groups: {"all": […], "web": […], "ungrouped": […]} |
| ansible\_check\_mode | A boolean that is true when running in check mode (see [“Check Mode”](https://www.safaribooksonline.com/library/view/Ansible:+Up+and+Running,+2nd+Edition/9781491979792/ch16.html" \l "check_mode)) |
| ansible\_play\_batch | A list of the inventory hostnames that are active in the current batch (see [“Running on a Batch of Hosts at a Time”](https://www.safaribooksonline.com/library/view/Ansible:+Up+and+Running,+2nd+Edition/9781491979792/ch09.html" \l "batch_of_hosts)) |
| ansible\_play\_hosts | A list of all of the inventory hostnames that are active in the current play |
| ansible\_version | A dict with Ansible version info: {"full": 2.3.1.0", "major": 2, "minor": 3, "revision": 1, "string": "2.3.1.0"} |
| *Table 4-1. Built-in variables* | |

Некоторые полезные переменные:

- stdout\_callback = debug – для красивого показа ошибок и другой инфы (определяется в секции defaults);

**Inventory file**

Это ини файл.

Cодержит хосты, о которых ансибл будете знать, для каждого хоста в одну строку указывается:

алиас, ip адрес, порт, ансибл пользователь и ансибл приватный ключ.

Пример: testserver ansible\_host=127.0.0.1 ansible\_port=2222 ansible\_user=vagrant ansible\_private\_key\_file=.vagrant/machines/default/virtualbox/private\_key

Параметры использумые для хостов:

|  |  |  |
| --- | --- | --- |
| ansible\_host | Name of host | Hostname or IP address to SSH to |
| ansible\_port | 22 | Port to SSH to |
| ansible\_user | Root | User to SSH as |
| ansible\_password | (*None*) | Password to use for SSH authentication |
| ansible\_connection | smart | How Ansible will connect to host (see the following section) |
| ansible\_private\_key\_file | (*None*) | SSH private key to use for SSH authentication |
| ansible\_shell\_type | sh | Shell to use for commands (see the following section) |
| ansible\_python\_interpreter | */usr/bin/python* | Python interpreter on host (see the following section) |
| ansible\_\*\_interpreter | (*None*) | Like ansible\_python\_interpreter for other languages (see the following section) |

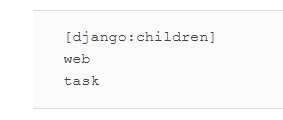
Группы хостов

Ансибл по умолчанию определяет группу All для всех хостов

Имя группы является атрибутом - [GroupName]

При этом под именем группы могут располагаться как определения хостов так и просто алиасы уже определенных выше хостов.

Группа включающая группы:



Группа может включать множество хостов имеющих какой-то темплейт в имени, где различается только номер (в примере будет 20 хостов начиная с web1.example.com):

[web]

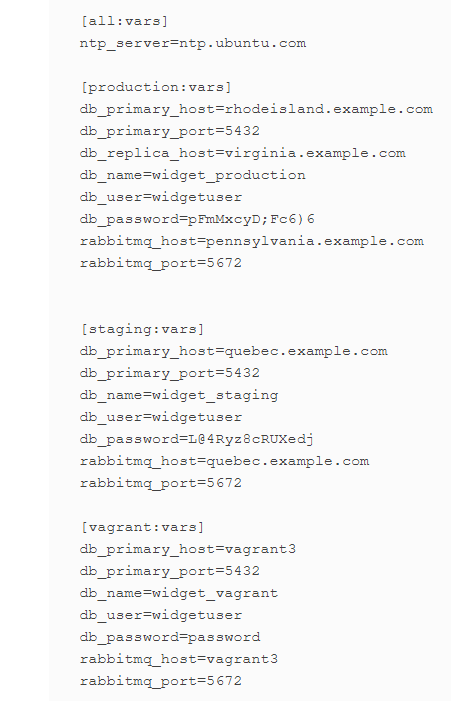
web[1:20].example.com

В общем есть множество других паттернов, по которым можно задавать хосты в группах и в секции hosts плейбуки:

| **Action** | **Example usage** |
| --- | --- |
| All hosts | all |
| All hosts | \* |
| Union | dev:staging |
| Intersection | staging:&database |
| Exclusion | dev:!queue |
| Wildcard | \*.example.com |
| Range of numbered servers | web[5:10] |
| Regular expression | ~web\d+\.example\.(com|org) |

Ansible supports multiple combinations of patterns—for example:

hosts: dev:staging:&database:!queue



Переменные можно вынести в отдельный файл, при этом для хостов нужно создать директорию host\_vars а для групп group\_vars. В этих директориях будут лежать yaml файлы с названиями хостов или групп, в которых будут определены переменны. Папки host\_vars и group\_vars должны быть в папке с плейбуками или инвентори файлом.

db:

user: widgetuser

password: pFmMxcyD;Fc6)6

name: widget\_production

primary:

host: rhodeisland.example.com

port: 5432

replica:

host: virginia.example.com

port: 5432

rabbitmq:

host: pennsylvania.example.com

port: 5672

**Modules**

apt

Installs or removes packages by using the apt package manager

copy

Copies a file from local machine to the hosts

file

Sets the attribute of a file, symlink, or directory

service

Starts, stops, or restarts a service

template

Generates a file from a template and copies it to the hosts

add\_host name=hostname groups=web,staging myvar=myval

Add new host to inventory file

Wait\_for

Wait for respond of remote host

- name: wait for ssh server to be running

local\_action: wait\_for port=22 host="{{ inventory\_hostname }}"

search\_regex=OpenSSH

Для просмотра информации по модулю, можно воспользоваться командой:

ansible-doc *<module-name>*

**Roles**

Ролью могут наделяться несколько хостов. Структура:

roles/database/tasks/main.yml - **Tasks**

roles/database/files/ - **Holds files to be uploaded to hosts**

roles/database/templates/ - **Holds Jinja2 template files**

roles/database/handlers/main.yml - **Handlers**

roles/database/vars/main.yml - **Variables that shouldn’t be overridden**

roles/database/defaults/main.yml - **Default variables that can be overridden**

roles/database/meta/main.yml - **Dependency information about a role**

При это не все папки обязательно должны быть

Для изменения пути по которому искать роли, можно прописать переменную в ansible.cfg:

**[defaults]**

roles\_path = ~/ansible\_roles

Зависимые роли

Это роль, которая автоматически применяется, если применена какая-то другая роль/роли.

Чтобы указать что у роли (например web) есть зависимая роль, нужно создать такой файл:

roles/web/meta/main.yml

dependencies:

- { role: ntp, ntp\_server=ntp.ubuntu.com }

В файле указывается имя роли и задаются нужные параметры для этой роли. Можно указывать в этом файле не одну а несколько ролей.

Опен сорсные роли можно найти здесь: <https://galaxy.ansible.com/>

Чтобы применить роль к плейбуке нужно прописать переменную:

- name: deploy mezzanine on vagrant

hosts: web

vars\_files:

- secrets.yml

roles:

- role: database

database\_name: "{{ mezzanine\_proj\_name }}"

database\_user: "{{ mezzanine\_proj\_name }}"

- role: mezzanine

live\_hostname: 192.168.33.10.xip.io

domains:

- 192.168.33.10.xip.io

- www.192.168.33.10.xip.io

А чтобы применить часть роли, к конкретной части плейбуки, нужно применять include\_role, например к таске плейбуки:

- name: install php

include\_role:

name: php [1](https://www.safaribooksonline.com/library/view/ansible-up-and/9781491979792/ch08.html#callout_complex_playbooks_CO5-1)

А вот запуск определенного файла(install.yml) тасков роли из таски плейбуки:

- name: install php

include\_role:

name: php

tasks\_from: install [1](https://www.safaribooksonline.com/library/view/ansible-up-and/9781491979792/ch08.html#callout_complex_playbooks_CO6-1)

**Filters**

Default

если левая часть (переменная) не установлена, то возвращается правая часть:

"HOST": "{{ database\_host | default('localhost') }}",

Task Result

сверяет статус таска с заданным (failed,changed,success,skipped)и возвращает true/false:

name: Run myprog

command: /opt/myprog

register: result

ignore\_errors: True

- debug: var=result

debug: msg="Stop running the playbook if myprog failed"

failed\_when: result|failed

Фильтры путей

Применяется к переменным содержащие путь к файлу

|  |  |
| --- | --- |
| basename | Base name of file path |
| dirname | Directory of file path |
| expanduser | File path with ~ replaced by home directory |
| realpath | Canonical path of file path, resolves symbolic links |

Join

Объединяет список элементов:

ALLOWED\_HOSTS = [{{ domains|join(", ") }}]

**Lookups**

По сути олицетворяет полезную функцию для прочтения файла, генерации пароля, получения результата некоторой команды и т.п..

| **Name** | **Description** |
| --- | --- |
| file | Contents of a file |
| password | Randomly generate a password |
| pipe | Output of locally executed command |
| env | Environment variable |
| template | Jinja2 template after evaluation |
| csvfile | Entry in a *.csv* file |
| dnstxt | DNS TXT record |
| redis\_kv | Redis key lookup |
| etcd | etcd key lookup |

lookup('file', '/path/to/file.txt')

Юзается в фигурных скобках везде, даже в j2.

Pipe

- name: get SHA of most recent commit

debug: msg="{{ lookup('pipe', 'git rev-parse HEAD') }}"

**Loops**

| **Name** | **Input** | **Looping strategy** |
| --- | --- | --- |
| with\_items | List | Loop over list elements |
| with\_lines | Command to execute | Loop over lines in command output |
| with\_fileglob | Glob | Loop over filenames |
| with\_first\_found | List of paths | First file in input that exists |
| with\_dict | Dictionary | Loop over dictionary elements |
| with\_flattened | List of lists | Loop over flattened list |
| with\_indexed\_items | List | Single iteration |
| with\_nested | List | Nested loop |
| with\_random\_choice | List | Single iteration |
| with\_sequence | Sequence of integers | Loop over sequence |
| with\_subelements | List of dictionaries | Nested loop |
| with\_together | List of lists | Loop over zipped list |
| with\_inventory\_hostnames | Host pattern | Loop over matching hosts |
| *Table 8-4. Looping constructs* | | |

Можно делать даже вложенные циклы, назначая переменную итератор (vhost.domain):

- name: run a set of tasks in one loop

include: vhosts.yml

with\_items:

- { domain: www1.example.com }

- { domain: www2.example.com }

- { domain: www3.example.com }

loop\_control:

loop\_var: vhost [1](https://www.safaribooksonline.com/library/view/Ansible:+Up+and+Running,+2nd+Edition/9781491979792/ch08.html#callout_complex_playbooks_CO1-1)

Где файл vhosts.yml:

- name: create nginx directories

file:

path: /var/www/html/{{ vhost.domain }}/{{ item }} [1](https://www.safaribooksonline.com/library/view/Ansible:+Up+and+Running,+2nd+Edition/9781491979792/ch08.html#callout_complex_playbooks_CO2-1)

state: directory

with\_items:

- logs

- public\_http

- public\_https

- includes

- name: create nginx vhost config

template:

src: "{{ vhost.domain }}.j2"

dest: /etc/nginx/conf.d/{{ vhost.domain }}.conf

Также мы можем вешать лейблы, чтобы не засорять лишними данными результат выполнения таски в консоли:

- name: create nginx vhost configs

template:

src: "{{ item.domain }}.conf.j2"

dest: "/etc/nginx/conf.d/{{ item.domain }}.conf"

with\_items:

- { domain: www1.example.com, ssl\_enabled: yes }

- { domain: www2.example.com }

- { domain: www3.example.com,

aliases: [ edge2.www.example.com, eu.www.example.com ] }

loop\_control:

label: "for domain {{ item.domain }}" [1](https://www.safaribooksonline.com/library/view/Ansible:+Up+and+Running,+2nd+Edition/9781491979792/ch08.html#callout_complex_playbooks_CO3-1)

[1](https://www.safaribooksonline.com/library/view/Ansible:+Up+and+Running,+2nd+Edition/9781491979792/ch08.html#co_complex_playbooks_CO3-1)

Результат будет выведен такой:

TASK [create nginx vhost configs] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [localhost] => (item=for domain www1.example.com)

ok: [localhost] => (item=for domain www2.example.com)

ok: [localhost] => (item=for domain www3.example.com)

**When**

Если надо выполнить таску, только если выполняется условие, применяется оператор when:

- name: install nginx

package:

name: nginx

tags: nginx

become: yes

when: ansible\_os\_family == 'RedHat'

Также можно применять условие и к нескольким таскам, обрамляя их в блок:

- block:

- name: install nginx

package:

name: nginx

- name: ensure nginx is running

service:

name: nginx

state: started

enabled: yes

become: yes

when: "ansible\_os\_family == 'RedHat'"

С блоками тесно связана обработка ошибок.

**Error handling**

Любые ошибки произошедшие в тасках обернутых в блоке можно обработать:

- block: [1](https://www.safaribooksonline.com/library/view/ansible-up-and/9781491979792/ch08.html#callout_complex_playbooks_CO7-1)

- debug: msg="You will see a failed tasks right after this"

- command: /bin/false

- debug: "You won't see this message"

rescue: [2](https://www.safaribooksonline.com/library/view/ansible-up-and/9781491979792/ch08.html#callout_complex_playbooks_CO7-2)

- debug: "You only see this message in case of an failure in the block"

always: [3](https://www.safaribooksonline.com/library/view/ansible-up-and/9781491979792/ch08.html#callout_complex_playbooks_CO7-3)

- debug: "This will be always executed"

**2 –** по сути catch, т.е. здесь выполняются таски если произошла ошибка. Только если ошибка случилась, то НИКАКИЕ таски после блока выполняться все равно не будут.

**3 –** по сути finally, таски которые выполнятся всегда.

**Includes**

Include позволяет дополнить таски из файла, их недостающими частями. Т.е. весь смысл в выносе одинаковых частей тасок в отдельный файл, чтобы уменьшить код.

*Example 8-25. Separate tasks into a different file*

- name: install nginx

package:

name: nginx

- name: ensure nginx is running

service:

name: nginx

state: started

enabled: yes

*Example 8-26. Using an include for the tasks file applying the arguments in common*

- include: nginx\_include.yml

tags: nginx

become: yes

when: ansible\_os\_family == 'RedHat'

Второй пример по сути означает две таски из первого примера, к каждой из которых применено tags, become и условие when.

**Debugging**

Если выставить в плейбуке стратегию debug, то ансибл будет врубать дебаггер каждый раз когда таска валится

| **Command** | **Description** |
| --- | --- |
| p var | Print out the value of a supported variable |
| task.args[key]=value | Modify an argument for the failed task |
| vars[key]=value | Modify the value of a variable |
| r | Rerun the failed task |
| c | Continue executing the play |
| q | Abort the play and execute the debugger |
| help | Show help message |
| *able 16-1. Debugger commands* | |

| **Command** | **Description** |
| --- | --- |
| p task | The name of the task that failed |
| p task.args | The module arguments |
| p result | The result returned by the failed task |
| p vars | Value of all known variables |
| p vars[key] | Value of one variable |
| *able 16-2. Variables supported by the debugger* | |

Ассерт

Валит плейбуку если не выполняется условие (Jinja2 синтаксис):

- name: assert that eth1 interface exists

assert:

that: ansible\_eth1 is defined

Статус файла можно прочекать с помощью модуля stat:

- name: stat /opt/foo

stat: path=/opt/foo

register: st

- name: assert that /opt/foo is a directory

assert:

that: st.stat.isdir

Он возвращает словарь с переменной stat и содержит следующую инфу:

| **Field** | **Description** |
| --- | --- |
| atime | Last access time of path, in Unix timestamp format |
| ctime | Creation time of path, in Unix timestamp format |
| dev | Numerical ID of the device that the inode resides on |
| exists | True if path exists |
| gid | Numerical group ID of path owner |
| inode | Inode number |
| isblk | True if path is block special device file |
| ischr | True if path is character special device file |
| isdir | True if path is a directory |
| isfifo | True if path is a FIFO (named pipe) |
| isgid | True if set-group-ID bit is set on file |
| islnk | True if path is a symbolic link |
| isreg | True if path is a regular file |
| issock | True if path is a Unix domain socket |
| isuid | True if set-user-ID bit is set on file |
| mode | File mode as a string, in octal (e.g., “1777”) |
| mtime | Last modification time of path, in Unix timestamp format |
| nlink | Number of hard links to the file |
| pw\_name | Login name of file owner |
| rgrp | True if group read permission enabled |
| roth | True if other read permission enabled |
| rusr | True if user read permission enabled |
| size | File size in bytes, if regular file |
| uid | Numerical user ID of path owner |
| wgrp | True if group write permission enabled |
| woth | True if other write permission enabled |
| wusr | True if user write permission enabled |
| xgrp | True if group execute permission enabled |
| xoth | True if other execute permission enabled |
| xusr | True if user execute permission enabled |

Перед тем как запустить плейбуку можно проверить ее, стартанув ее с специальными флагами (команда ansible-playbook):

* --syntax-check – проверяет синтаксис (не исполняя плейбуки)
* --list-hosts – выводит всех хостов плейбуки (не выолняюю плейбуку) (валится если хостов нет)
* --list-tasks – выводит все таски плейбуки (тоже не исполняя плейбуки)
* --check – пробует выяснить, изменят ли таски состояние хоста
* **--**diff – показывает отличия для любого измененного файла на удаленной машине (можно использовать с флагом check)
* **--**step – ансибл будет выбрасывать попап перед тем как стартануть следующую таску
* --start-at-task –позволяет ансиблу начать плейбуку с определенной таски (nsible-playbook --start-at-task="install packages" playbook.yml)
* --tags – исполняет в плейбуке те таски и плеи, которые обозначены определенным тагом (ansible-playbook --tags=foo,bar playbook.yml).

**Вопросы**

- Факты это переменные которыми управляет сам ансибл? Т.е. это вся инфа которая автоматом собирается с хостов + результат некоторых модулей?

- Ansible.cfg – какое у него фактическое назначение? Глобальные переменные и глобальный оверрайдинг?

- ignore\_errors в таске предовтратит отработки rescue секции при эррор хэндлинге?