Internal training INTRODUCTION TO ANSIBLE

Author / manager: Lev Goncharov / Ilya Semerhanov

Lection #4 – Plugins & modules

ERLEBEN, WAS VERBINDET.

Course Content

1. Lection #1: Introduction
    1. Configuration management
    2. Ansible. How it works?
    3. Vagrant. Training env
2. Lection #2: First playbook
    1. First playbook
    2. Ansible modules
    3. Facts & variables
3. Lection #3: Base features
    1. Jinja2 templating
    2. Conditions
    3. Loops
4. Lection #4: Plugins & modules
    1. Plugins
    2. Modules
    3. Handlers
5. Lection #5: Best practices
    1. Roles
    2. Working with inventory
    3. Repository structure
6. Lection #6: Usecases
    1. Golden images
    2. Management infrastructure
    3. CI/CD integration

Course Content

1. Lection #1: Introduction
    1. Configuration management
    2. Ansible. How it works?
    3. Vagrant. Training env
2. Lection #2: First playbook
    1. First playbook
    2. Ansible modules
    3. Facts & variables
3. Lection #3: Base features
    1. Jinja2 templating
    2. Conditions
    3. Loops
4. Lection #4: Plugins & filters
    1. Plugins
    2. Modules
    3. Handlers
5. Lection #5: Best practices
    1. Roles
    2. Working with inventory
    3. Repository structure
6. Lection #6: Usecases
    1. Golden images
    2. Management infrastructure
    3. CI/CD integration

Lection #4. Plugins and modules
Plugins

- Pieces of code that augment Ansible’s core functionality
- Enable a rich, flexible and expandable feature set.
- Execute on ansible control host
- Run in current ansible process context
- https://docs.ansible.com/ansible/2.6/plugins/plugins.html

Lection #4. Plugins and modules

Plugins

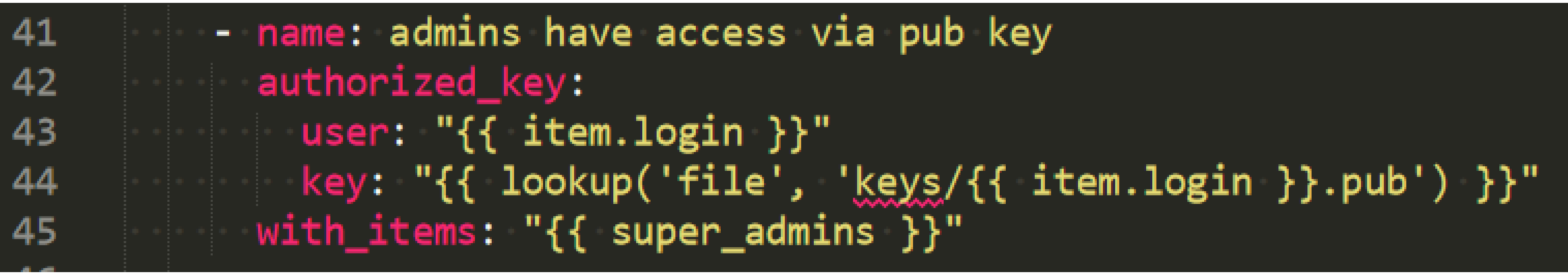
- Action Plugins
- Cache Plugins
- Callback Plugins
- Connection Plugins
- Inventory Plugins
- Lookup Plugins
- Shell Plugins
- Strategy Plugins
- Vars Plugins
- Filters
- Tests
- Plugin Filter Configuration

Lection #4. Plugins and modules
Plugins. Lookups plugins

allow access to outside data sources

https://docs.ansible.com/ansible/2.6/user_guide/playbooks_lookups.html

https://docs.ansible.com/ansible/2.6/plugins/lookup.html#plugin-list



```
    - name: admins have access via pub key
      authorized_key:
        user: "{{ item.login }}"
        key: "{{ lookup('file', 'keys/{{ item.login }}.pub') }}"
      with_items: "{{ super_admins }}"
```

Lection #4. Plugins and modules

Plugins. Lookups plugins

aws_account_attribute - Look up AWS account attributes.

aws_service_ip_ranges - Look up the IP ranges for services provided in AWS such as EC2 and S3.

aws_ssm - Get the value for a SSM parameter or all parameters under a path.

cartesian - returns the cartesian product of lists

chef_databag - fetches data from a Chef Databag

config - Lookup current Ansible configuration values

conjur_variable - Fetch credentials from CyberArk Conjur.

consul_kv - Fetch metadata from a Consul key value store.

credstash - retrieve secrets from Credstash on AWS

csvfile - read data from a TSV or CSV file

cyberarkpassword - get secrets from CyberArk AIM

dict - returns key/value pair items from dictionaries

dig - query DNS using the dnspython library

dnstxt - query a domain(s)'s DNS txt fields

env - read the value of environment variables

etcd - get info from etcd server

file - read file contents

fileglob - list files matching a pattern

filetree - recursively match all files in a directory tree

first_found - return first file found from list

flattened - return single list completely flattened

hashi_vault - retrieve secrets from HashiCorp's vault

hiera - get info from hiera data

indexed_items - rewrites lists to return 'indexed items'

ini - read data from a ini file

inventory_hostnames - list of inventory hosts matching a host pattern

items - list of items

k8s - Query the K8s API

keyring - grab secrets from the OS keyring

lastpass - fetch data from lastpass

lines - read lines from command

list - simply returns what it is given.

mongodb - lookup info from MongoDB

nested - composes a list with nested elements of other lists

nios - Query Infoblox NIOS objects

nios_next_ip - Return the next available IP address for a network

onepassword - fetch field values from 1Password

onepassword_raw - fetch raw json data from 1Password

password - retrieve or generate a random password, stored in a file

passwordstore - manage passwords with passwordstore.org's pass utility

pipe - read output from a command

random_choice - return random element from list

redis - fetch data from Redis

redis_kv - fetch data from Redis

sequence - generate a list based on a number sequence

shelvefile - read keys from Python shelve file

subelements - traverse nested key from a list of dictionaries

template - retrieve contents of file after templating with Jinja2

together - merges lists into syncronized list

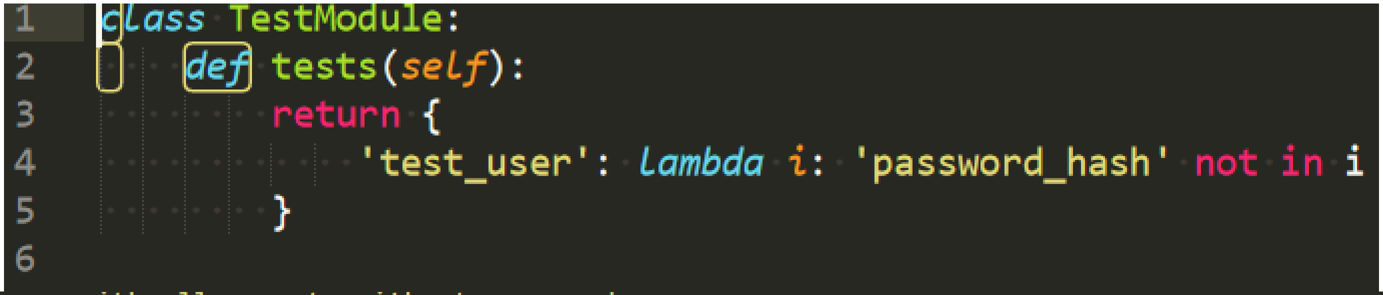
url - return contents from URL

vars - Lookup templated value of variables

Lection #4. Plugins and modules

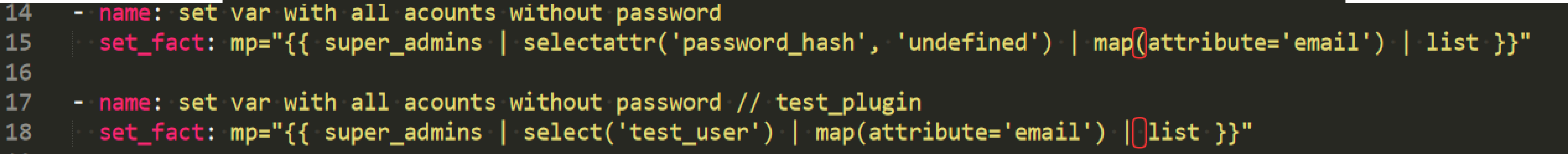
Plugins. Developing

- Hide dirty magic inside python
- Do you really need? Already exist?
- https://docs.ansible.com/ansible/2.6/dev_guide/developing_plugins.html



```
class TestModule:
    def tests(self):
        return {
            'test_user': lambda i: 'password_hash' not in i
        }

```



```
- name: set var with all acounts without password
  set_fact: mp="{{ super_admins | selectattr('password_hash', 'undefined') | map(attribute='email') | list }}"

- name: set var with all acounts without password // test_plugin
  set_fact: mp="{{ super_admins | select('test_user') | map(attribute='email') | list }}"
```

Course Content

1. Lection #1: Introduction
   1. Configuration management
   2. Ansible. How it works?
   3. Vagrant. Training env
2. Lection #2: First playbook
   1. First playbook
   2. Ansible modules
   3. Facts & variables
3. Lection #3: Base features
   1. Jinja2 templating
   2. Conditions
   3. Loops
4. Lection #4: Plugins & filters
   1. Plugins
   2. Modules
   3. Handlers
5. Lection #5: Best practices
   1. Roles
   2. Working with inventory
   3. Repository structure
6. Lection #6: Usecases
   1. Golden images
   2. Management infrastructure
   3. CI/CD integration

Lection #4. Plugins and modules
Modules

- Execute on target host
- Run in new context
- Pack in ansiballz before sending to target
- Accept parameters if needed(via parameters file)
- Print JSON result into STDOUT

Lection #4. Plugins and modules
Modules. How it works?

1. Ansible get next task from queue, detect module name
2. If action plugin with same name exist, run it
3. Prepare parameter file for module
4. Copy files to target host
   - AnsibleModule – ansiballz
   - Others – parameter file & executable
5. Run ansiballz or module
6. Module do something
7. Ansible get result in JSON

Lection #4. Plugins and modules

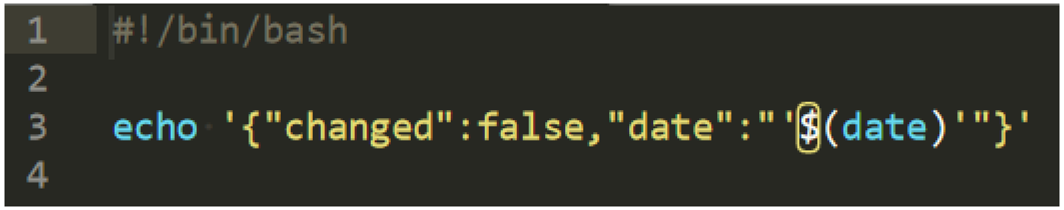
Modules. Standart

All modules
Cloud modules
Clustering modules
Commands modules
Crypto modules
Database modules
Files modules
Identity modules
Inventory modules
Messaging modules
Monitoring modules
Net Tools modules
Network modules
Notification modules
Packaging modules
Remote Management modules
Source Control modules
Storage modules
System modules
Utilities modules
Web Infrastructure modules
Windows modules

https://docs.ansible.com/ansible/latest/modules/modules_by_category.html

Lection #4. Plugins and modules
Modules. Developing

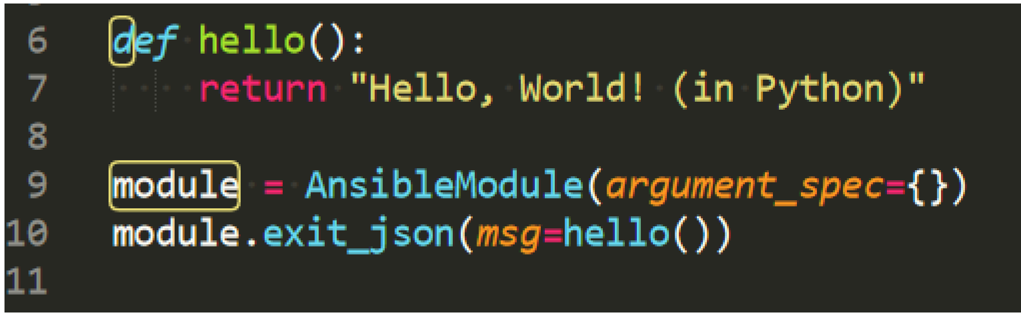
- Do you really need it? Already exist?
- Hide dirty magic
- Powershell / python / bash / binary / ....
- https://docs.ansible.com/ansible/2.6/dev_guide/developing_modules.html



```bash
#!/bin/bash

echo '{"changed":false,"date":"'$(date)'"}'

```



```python
def hello():
    return "Hello, World! (in Python)"

module = AnsibleModule(argument_spec={})
module.exit_json(msg=hello())

```

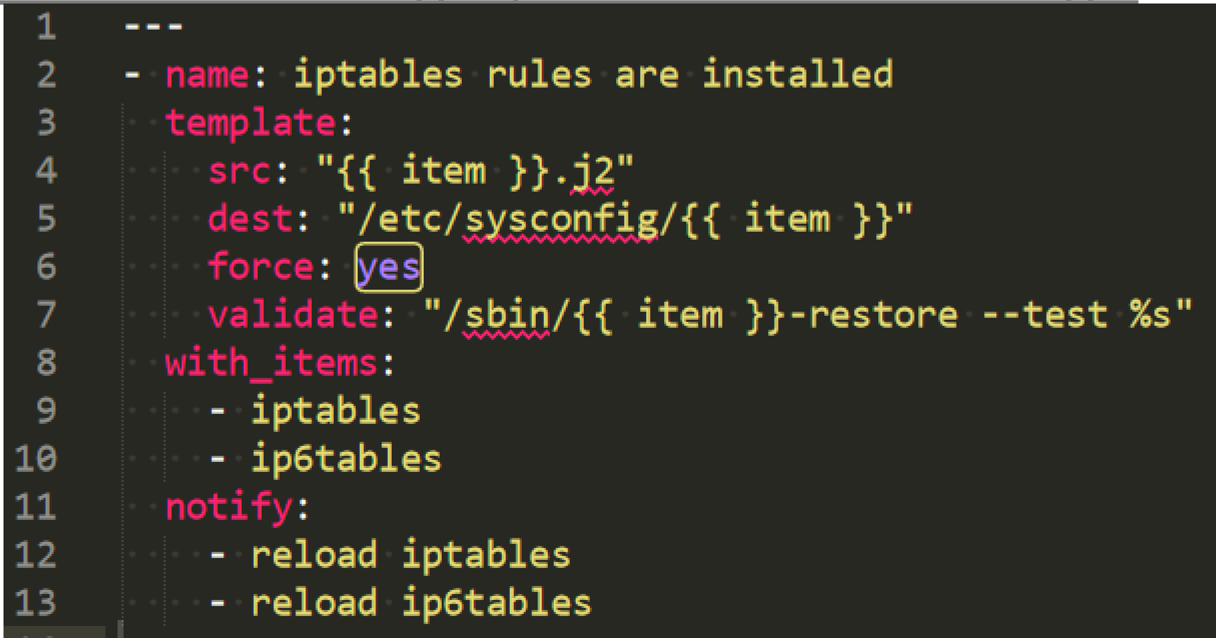
Course Content

1. Lection #1: Introduction
   1. Configuration management
   2. Ansible. How it works?
   3. Vagrant. Training env
2. Lection #2: First playbook
   1. First playbook
   2. Ansible modules
   3. Facts & variables
3. Lection #3: Base features
   1. Jinja2 templating
   2. Conditions
   3. Loops
4. Lection #4: Plugins & filters
   1. Plugins
   2. Modules
   3. Handlers
5. Lection #5: Best practices
   1. Roles
   2. Working with inventory
   3. Repository structure
6. Lection #6: Usecases
   1. Golden images
   2. Management infrastructure
   3. CI/CD integration

Lection #4. Plugins and modules

Handlers

- Triggered at the end of each block of tasks in a play
- Triggered once if notified by multiple different tasks.
- Name must be uniq
- https://docs.ansible.com/ansible/2.6/user_guide/playbooks_intro.html#handlers-running-operations-on-change



```yaml
---
- name: iptables rules are installed
  template:
    src: "{{ item }}.j2"
    dest: "/etc/sysconfig/{{ item }}"
    force: yes
    validate: "/sbin/{{ item }}-restore --test %s"
  with_items:
    - iptables
    - ip6tables
  notify:
    - reload iptables
    - reload ip6tables
```

ERLEBEN, WAS VERBINDET.

Lection #4. Plugins and modules
Workshop

```
$env:http_proxy='http://spbsrv-proxy2.t-systems.ru:3128'

$env:https_proxy='http://spbsrv-proxy2.t-systems.ru:3128'

git clone http://projects.t-systems.ru/lgonchar/ansible_course

cd student_files/04

vagrant up –provider hyperv
```

Lection #4. Plugins and modules
Workshop

```
$env:http_proxy='http://spbsrv-proxy2.t-systems.ru:3128'
$env:https_proxy='http://spbsrv-proxy2.t-systems.ru:3128'
git clone http://projects.t-systems.ru/lgonchar/ansible-course-public.git
cd student_files/04
vagrant up –provider hyperv
```

Lection #4. Plugins and modules

Homework

Modify existing playbook:

- Write simple bash/python module
  - It should wrap reload iptables
- Modify existing "reload iptables" handler. It should use the module from previous step

THANK YOU!

Q&A

"

Use the ansible, Luke

"

Obi Wan Kenobi

lev.goncharov@t-systems.com

ERLEBEN, WAS VERBINDET.