Internal training
INTRODUCTION TO ANSIBLE

Author / manager: Lev Goncharov / Ilya Semerhanov

Lection #3 – Base features

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

Lection #3. Base features
Including and importing

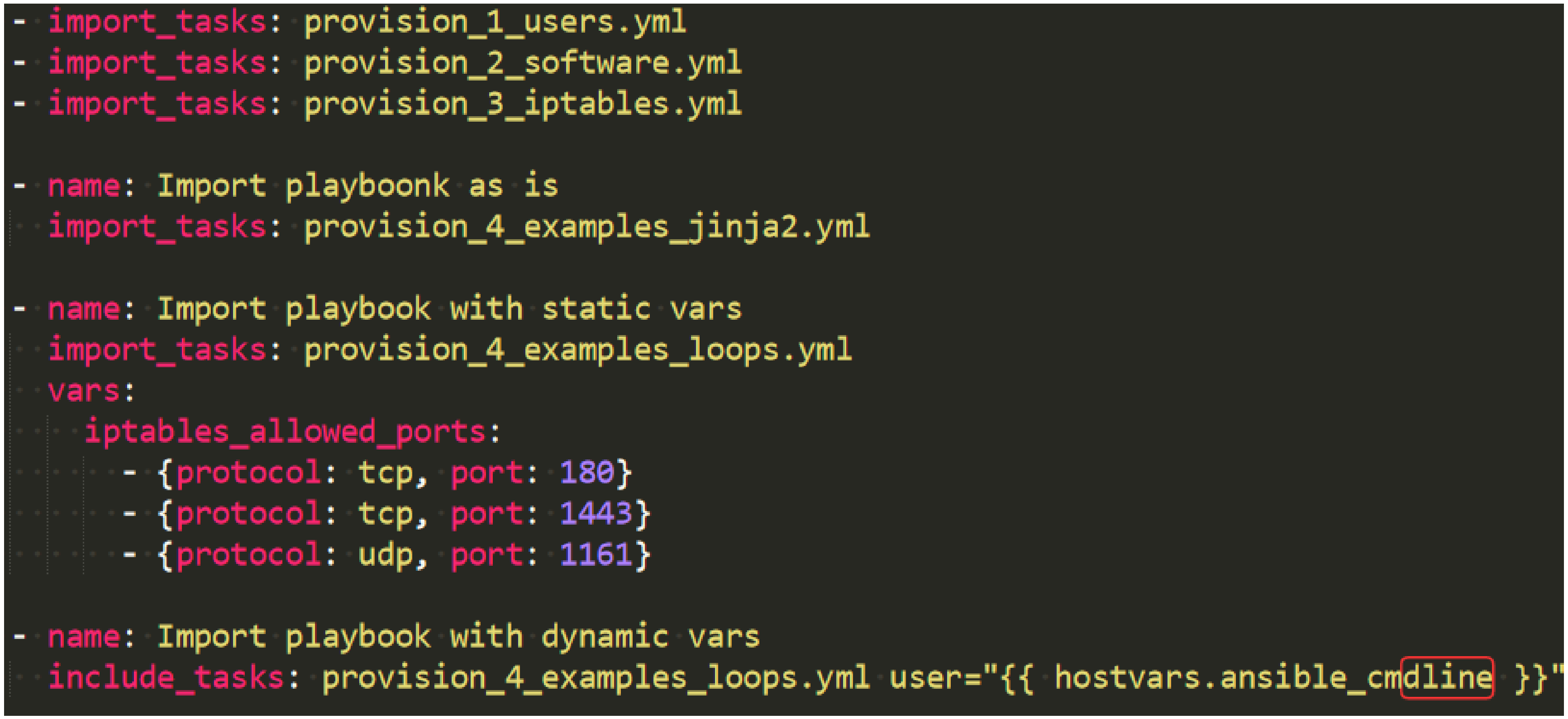
- import* - are pre-processed at the time playbooks are parsed.
- include* - are pre-processed as they encountered during the execution of the playbook

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

Lection #3. Base features

Including and importing

- import* - Use when you deal with logical "units".
- include* - Use to deal with different workflows and take decisions based on some dynamically gathered facts



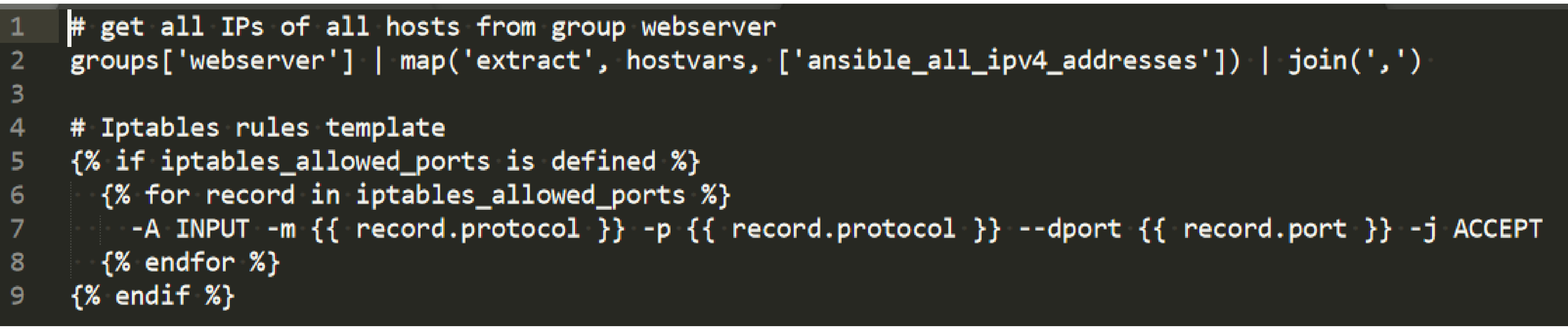
```yaml
- import_tasks: provision_1_users.yml
- import_tasks: provision_2_software.yml
- import_tasks: provision_3_iptables.yml

- name: Import playboonk as is
  import_tasks: provision_4_examples_jinja2.yml

- name: Import playbook with static vars
  import_tasks: provision_4_examples_loops.yml
  vars:
    iptables_allowed_ports:
      - {protocol: tcp, port: 180}
      - {protocol: tcp, port: 1443}
      - {protocol: udp, port: 1161}

- name: Import playbook with dynamic vars
  include_tasks: provision_4_examples_loops.yml user="{{ hostvars.ansible_cmdline }}"
```

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

Lection #3. Base features
Jinja2

- Templating language for Python
- Filters/tests/loops …
- http://jinja.pocoo.org/docs/2.10/



```
# get all IPs of all hosts from group webserver
groups['webserver'] | map('extract', hostvars, ['ansible_all_ipv4_addresses']) | join(',')

# Iptables rules template
{% if iptables_allowed_ports is defined %}
  {% for record in iptables_allowed_ports %}
    -A INPUT -m {{ record.protocol }} -p {{ record.protocol }} --dport {{ record.port }} -j ACCEPT
  {% endfor %}
{% endif %}
```

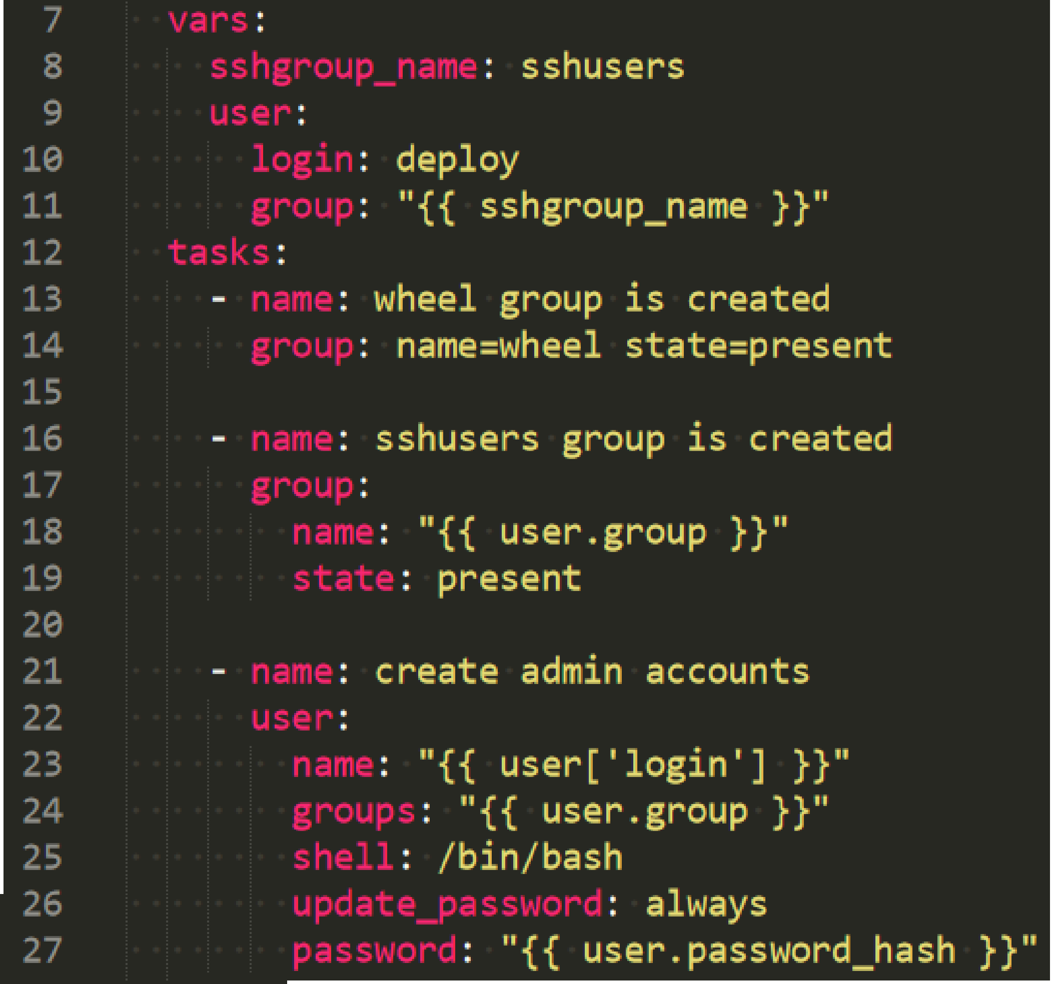
Lection #3. Base features
Jinja2 templating

- Variables
- Tests
- Filters
- Lookups

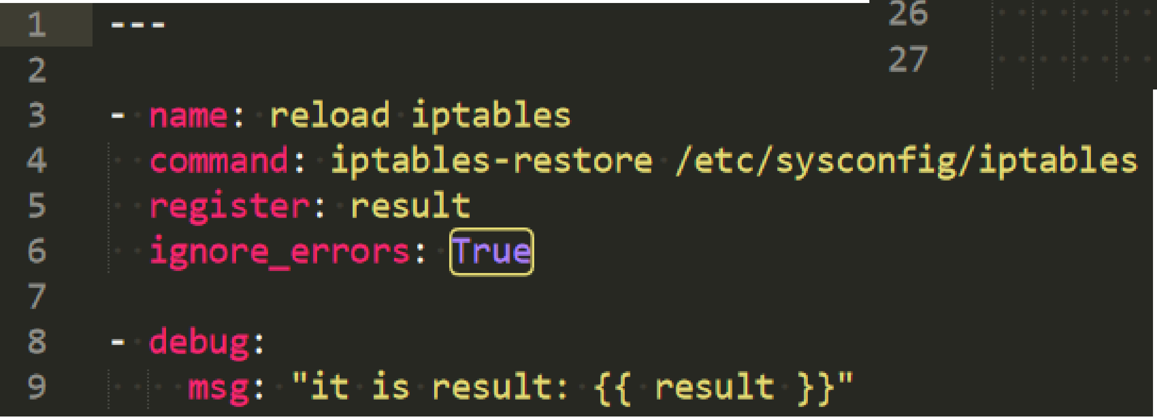
All templating happens on the Ansible controller before the task is sent and executed on the target machine

Lection #3. Base features
Jinja2 templating. Variables

Ansible allows you to reference variables in your playbooks using the Jinja2 templating system



```yaml
  vars:
    sshgroup_name: sshusers
    user:
      login: deploy
      group: "{{ sshgroup_name }}"
  tasks:
    - name: wheel group is created
      group: name=wheel state=present

    - name: sshusers group is created
      group:
        name: "{{ user.group }}"
        state: present

    - name: create admin accounts
      user:
        name: "{{ user['login'] }}"
        groups: "{{ user.group }}"
        shell: /bin/bash
        update_password: always
        password: "{{ user.password_hash }}"
```



```yaml
---

- name: reload iptables
  command: iptables-restore /etc/sysconfig/iptables
  register: result
  ignore_errors: True

- debug:
    msg: "it is result: {{ result }}"
```

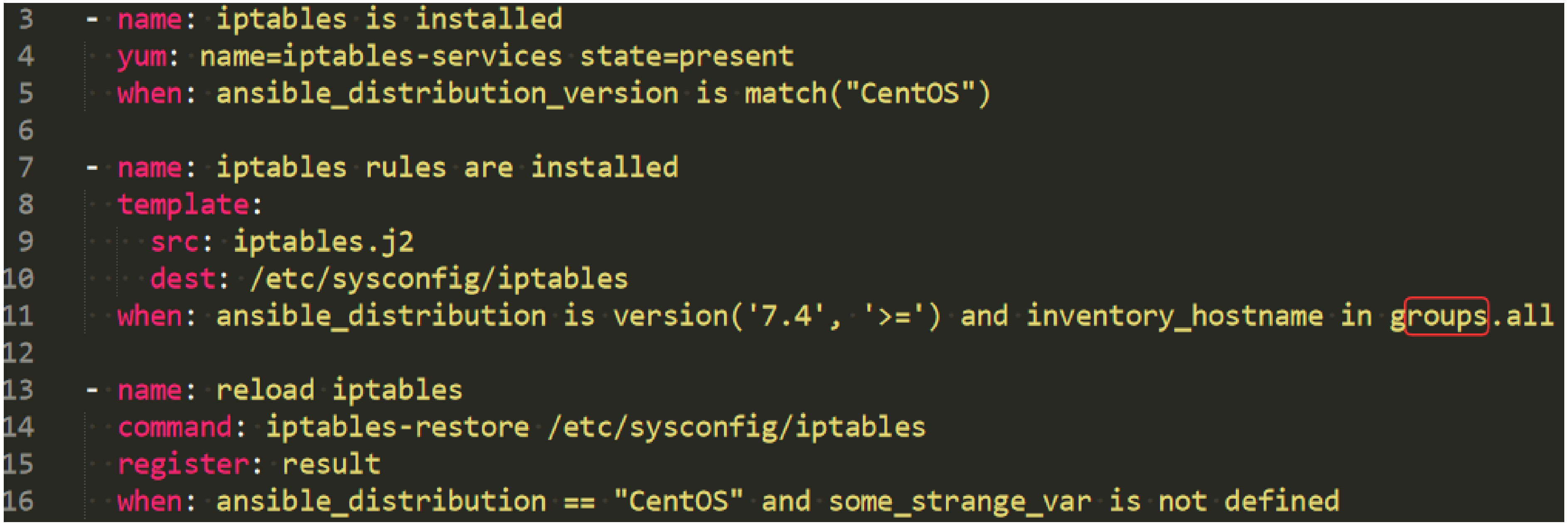
Lection #3. Base features
Jinja2 templating. Tests

- Way of evaluating template expressions and returning True or False
- https://docs.ansible.com/ansible/2.6/user_guide/playbooks_tests.html
- http://jinja.pocoo.org/docs/dev/templates/#tests

Lection #3. Base features

Jinja2 templating. Tests

- Strings match/search regex
- Version comparison
- Task results



```
- name: iptables is installed
  yum: name=iptables-services state=present
  when: ansible_distribution_version is match("CentOS")

- name: iptables rules are installed
  template:
    src: iptables.j2
    dest: /etc/sysconfig/iptables
  when: ansible_distribution is version('7.4', '>=') and inventory_hostname in groups.all

- name: reload iptables
  command: iptables-restore /etc/sysconfig/iptables
  register: result
  when: ansible_distribution == "CentOS" and some_strange_var is not defined
```

Lection #3. Base features
Jinja2 templating. Filters

transforming data inside a template expression.

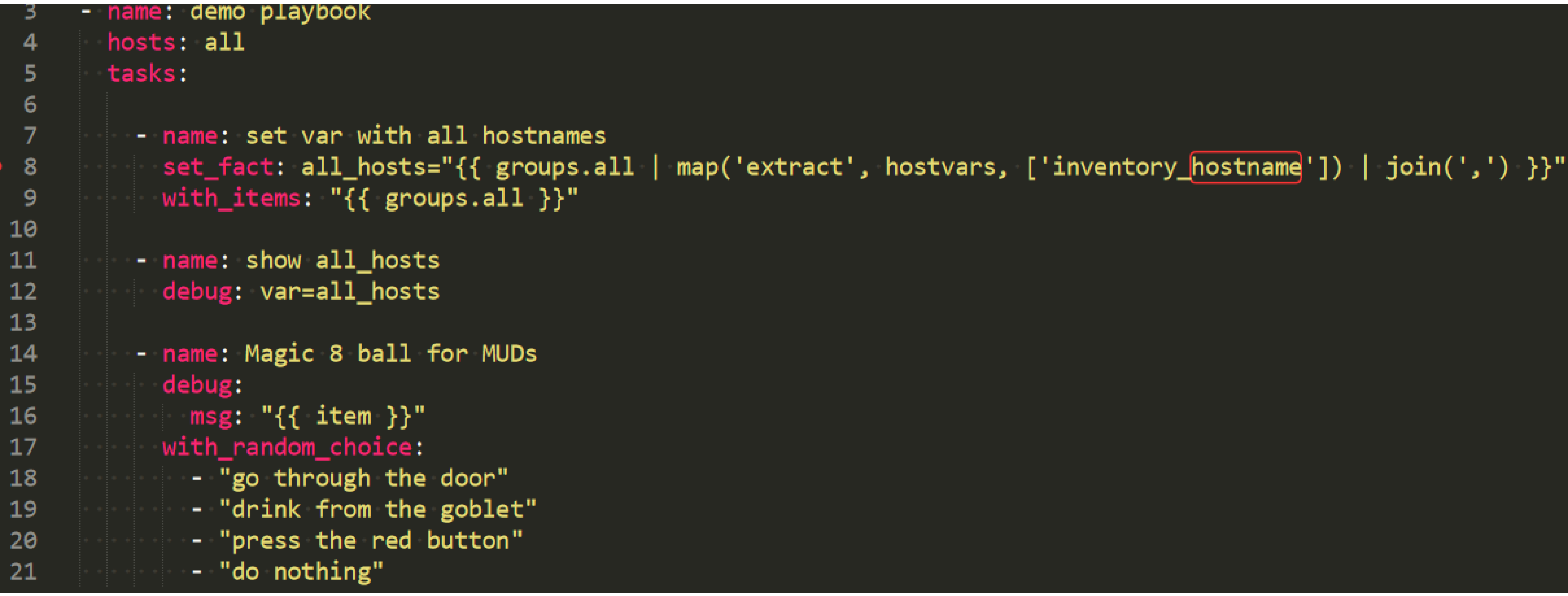
https://docs.ansible.com/ansible/2.6/user_guide/playbooks_filters.html

http://jinja.pocoo.org/docs/2.10/templates/#builtin-filters

Lection #3. Base features

Jinja2 templating. Filters

ansible-playbook -c local -i inventory.ini example.yml

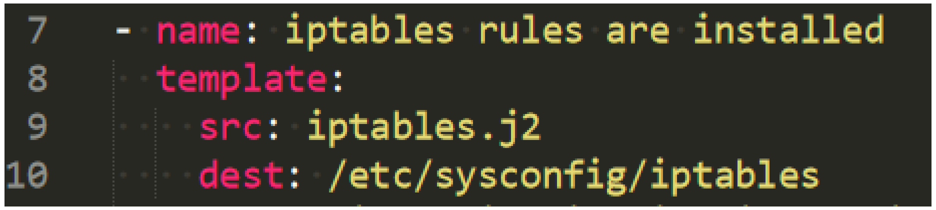


```
- name: demo playbook
  hosts: all
  tasks:

    - name: set var with all hostnames
      set_fact: all_hosts="{{ groups.all | map('extract', hostvars, ['inventory_hostname']) | join(',') }}"
      with_items: "{{ groups.all }}"

    - name: show all_hosts
      debug: var=all_hosts

    - name: Magic 8 ball for MUDs
      debug:
        msg: "{{ item }}"
      with_random_choice:
        - "go through the door"
        - "drink from the goblet"
        - "press the red button"
        - "do nothing"
```

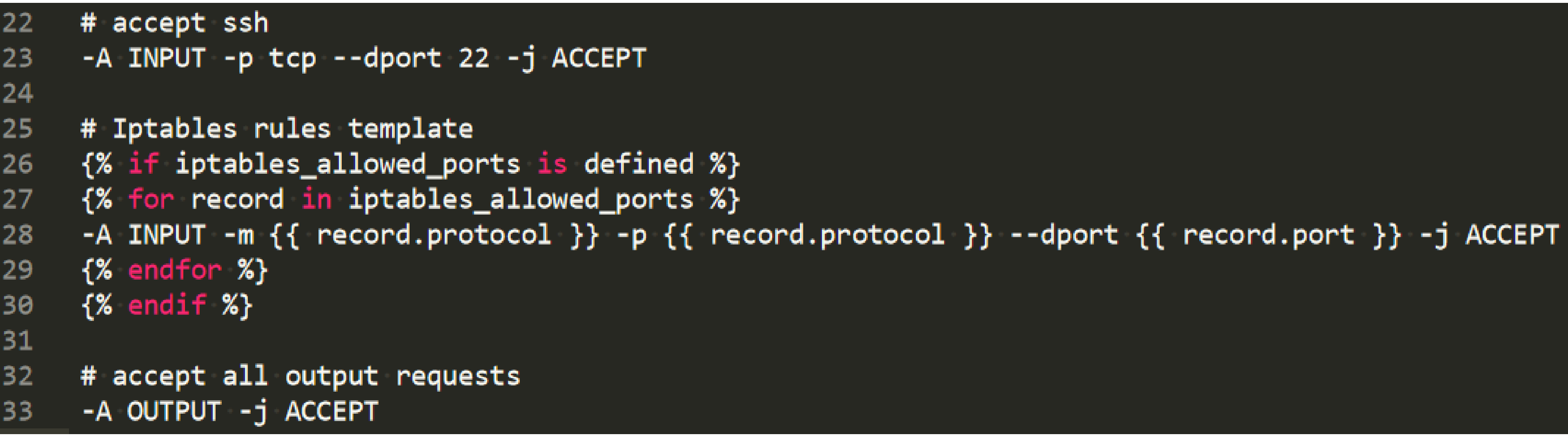
Lection #3. Base features

Jinja2 templating. Template module

Copy template from controller to target host



```
- name: iptables rules are installed
  template:
    src: iptables.j2
    dest: /etc/sysconfig/iptables
```



```
# accept ssh
-A INPUT -p tcp --dport 22 -j ACCEPT

# Iptables rules template
{% if iptables_allowed_ports is defined %}
{% for record in iptables_allowed_ports %}
-A INPUT -m {{ record.protocol }} -p {{ record.protocol }} --dport {{ record.port }} -j ACCEPT
{% endfor %}
{% endif %}

# accept all output requests
-A OUTPUT -j ACCEPT
```

Course Content

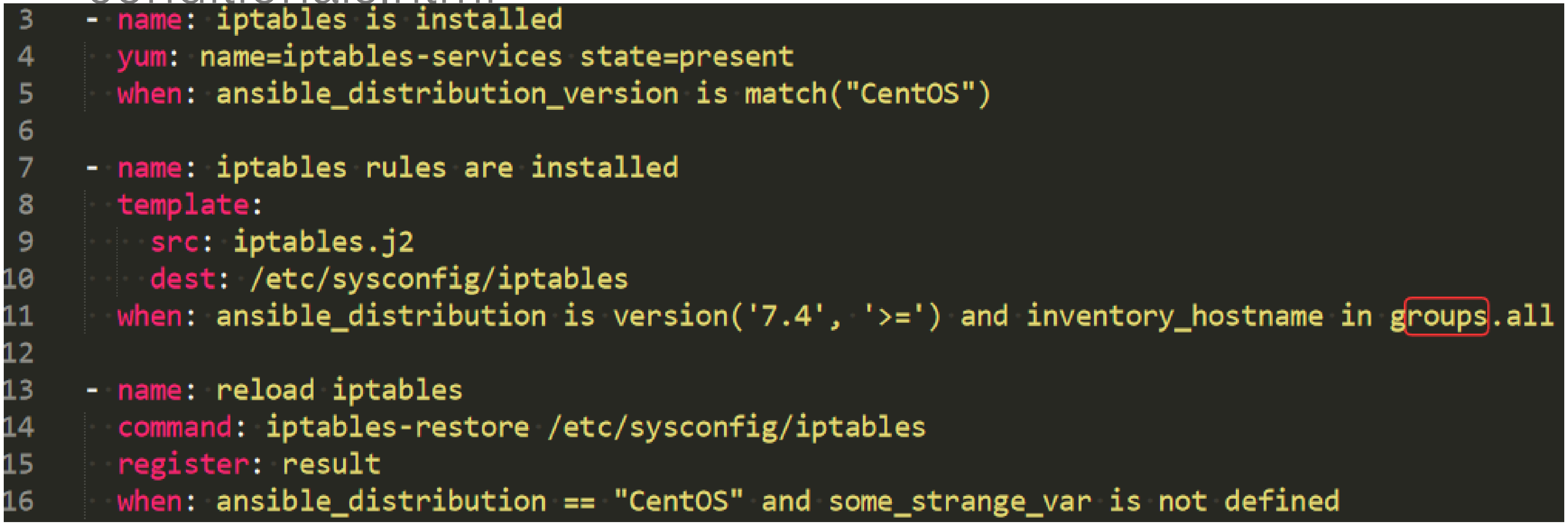
1. Lection #1: Introduction
   1. Configuration management
   2. Ansible. How it works?
   3. Vagrant. Training env
2. Lection #2: First playbook
   1. First playbook
   2. Ansible modules
   3. Facts & variables
3. Lection #3: Base features
   1. Jinja2 templating
   2. Conditions
   3. Loops
4. Lection #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

Lection #3. Base features

Conditions

- To skip a particular step on a particular host
- Use tests from jinja2 templates

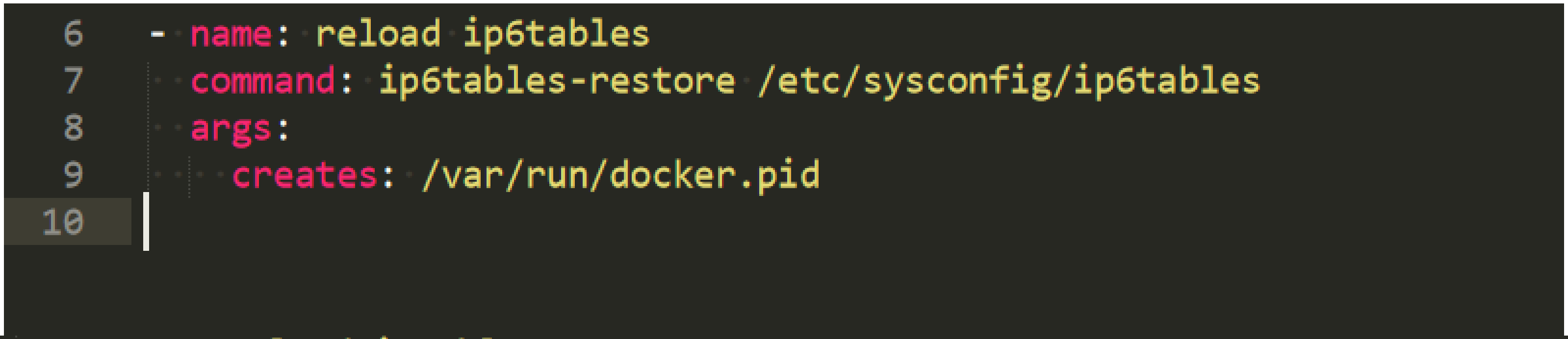
https://docs.ansible.com/ansible/2.6/user_guide/playbooks conditionals.html



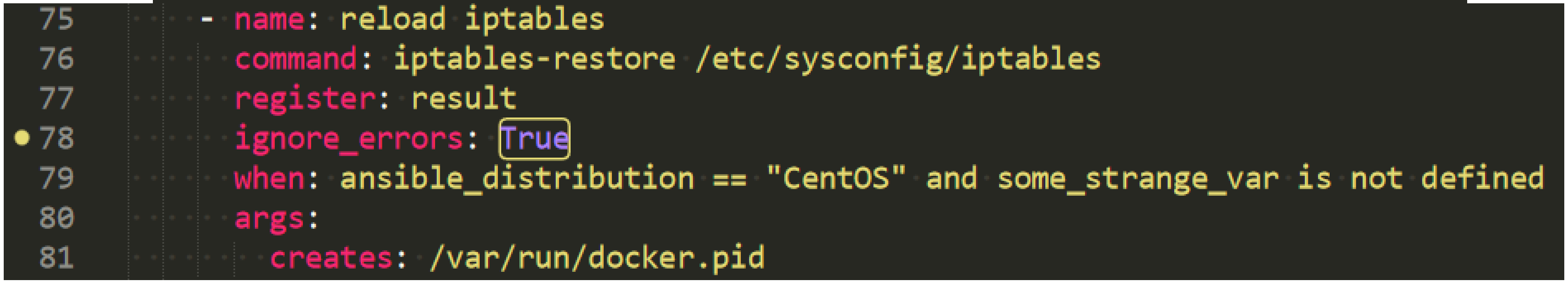
```yaml
- name: iptables is installed
  yum: name=iptables-services state=present
  when: ansible_distribution_version is match("CentOS")

- name: iptables rules are installed
  template:
    src: iptables.j2
    dest: /etc/sysconfig/iptables
  when: ansible_distribution is version('7.4', '>=') and inventory_hostname in groups.all

- name: reload iptables
  command: iptables-restore /etc/sysconfig/iptables
  register: result
  when: ansible_distribution == "CentOS" and some_strange_var is not defined
```

Lection #3. Base features
Conditions

- To skip a particular command if file exist
- Possible to combine with “when”



```
- name: reload ip6tables
  command: ip6tables-restore /etc/sysconfig/ip6tables
  args:
    creates: /var/run/docker.pid

```



```
    - name: reload iptables
      command: iptables-restore /etc/sysconfig/iptables
      register: result
      ignore_errors: True
      when: ansible_distribution == "CentOS" and some_strange_var is not defined
      args:
        creates: /var/run/docker.pid
```

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

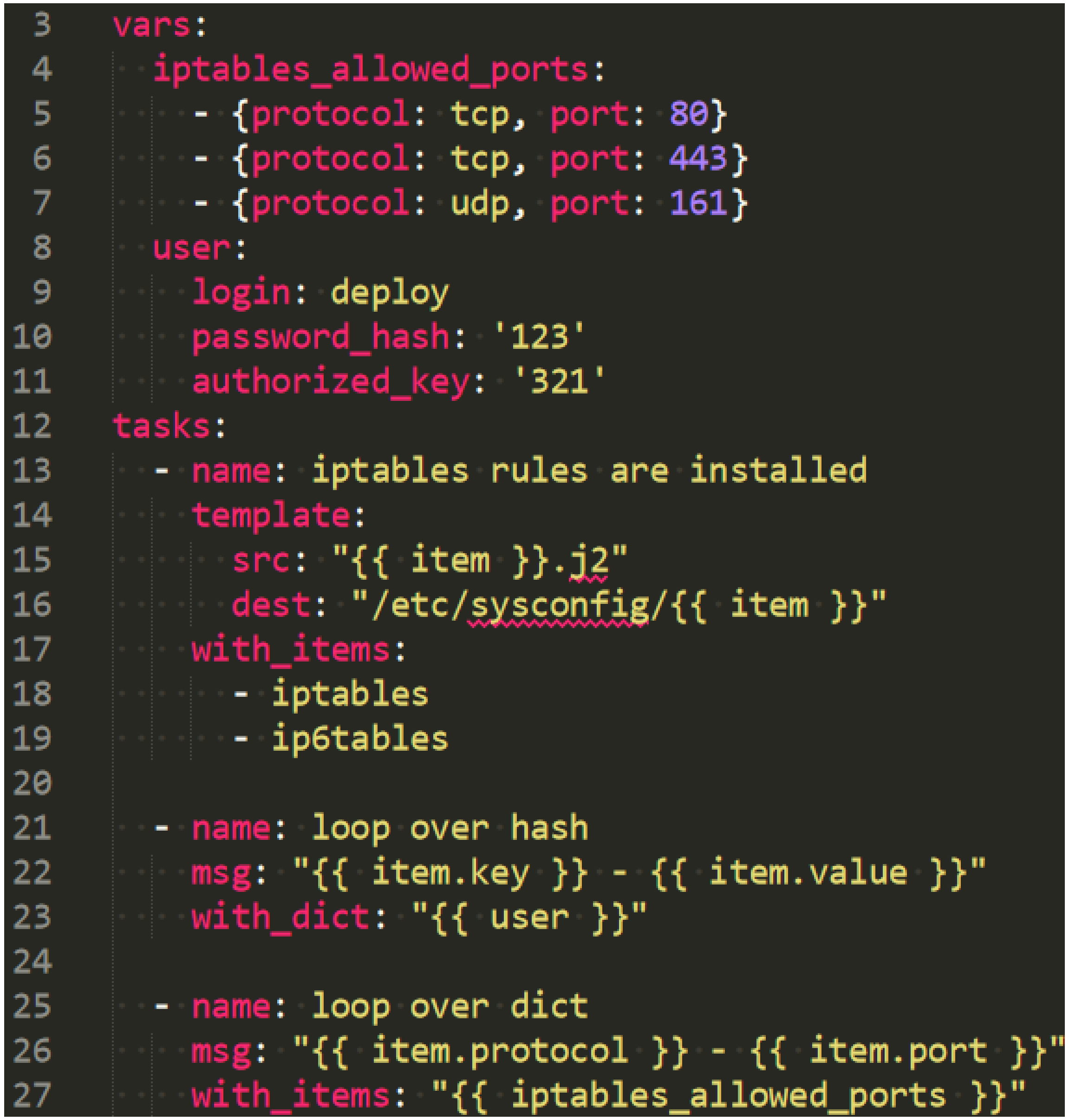
Lection #3. Base features
Loops

- Arrays
- Hashes
- Files
- Fileglobes
- Filetree
- Parallel Sets of Data
- Sequences
- Random choices
- Do-until loops

Lection #3. Base features

Loops

- Doc <= 2.4
- Doc >2.4

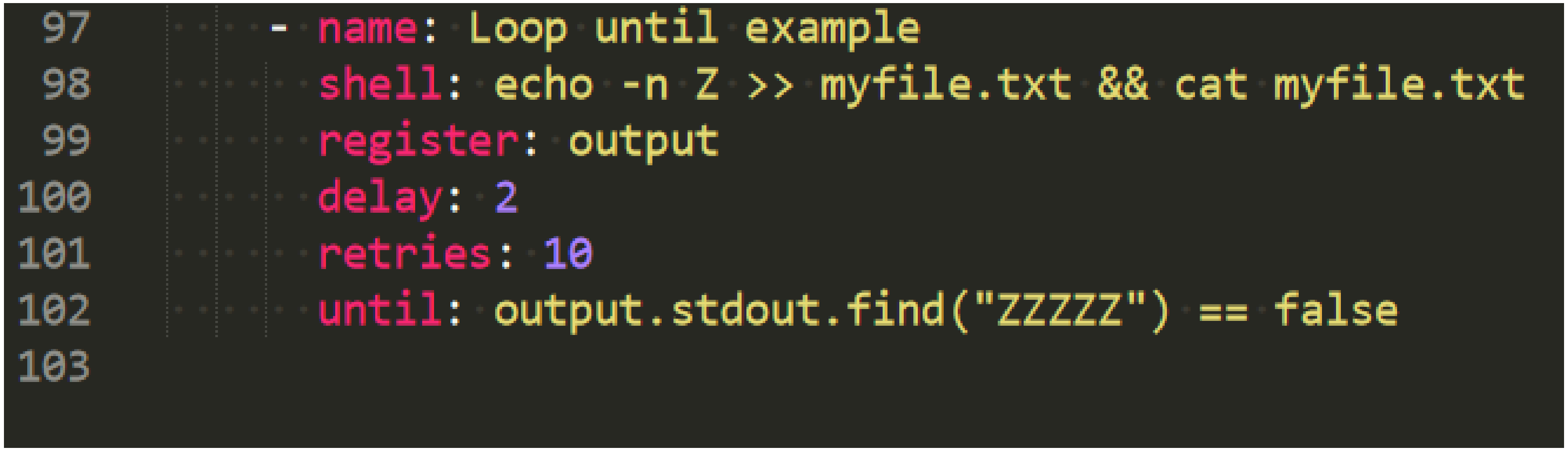


```
  vars:
    iptables_allowed_ports:
      - {protocol: tcp, port: 80}
      - {protocol: tcp, port: 443}
      - {protocol: udp, port: 161}
    user:
      login: deploy
      password_hash: '123'
      authorized_key: '321'
  tasks:
    - name: iptables rules are installed
      template:
        src: "{{ item }}.j2"
        dest: "/etc/sysconfig/{{ item }}"
      with_items:
        - iptables
        - ip6tables

    - name: loop over hash
      msg: "{{ item.key }} - {{ item.value }}"
      with_dict: "{{ user }}"

    - name: loop over dict
      msg: "{{ item.protocol }} - {{ item.port }}"
      with_items: "{{ iptables_allowed_ports }}"
```

Lection #3. Base features

Loops

- Register sets var each iterration



```
    - name: Loop until example
      shell: echo -n Z >> myfile.txt && cat myfile.txt
      register: output
      delay: 2
      retries: 10
      until: output.stdout.find("ZZZZZ") == false

```

Lection #3. Base features
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/03
vagrant up –provider hyperv
```

Lection #3. Base features
Homework

Modify existing playbook:

- Install snmpd (use loops)
- Configure snmpd via template module
  - Get snmp community string as from variable
- Open via template module & loops ports:
  - 161 udp
  - 443 tcp
- Generate self signed cert via openssl_certificate module
- Configure https for httpd
- Visit web site via https

THANK YOU!
Q&A

"

Use the ansible, Luke

"

Obi Wan Kenobi

lev.goncharov@t-systems.com

ERLEBEN, WAS VERBINDET.