Internal training
INTRODUCTION TO ANSIBLE

Author / manager: Lev Goncharov / Ilya Semerhanov

Lection #5 – Best practices

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 #5. Best practices
Ansible config

Certain settings in Ansible are adjustable via a configuration file. The stock configuration should be sufficient for most users, but there may be reasons you would want to change them.

https://raw.githubusercontent.com/ansible/ansible/devel/examples/ansible.cfg

Config processed in the following order:

1. ANSIBLE_CONFIG (an environment variable)
2. ansible.cfg (in the current directory)
3. .ansible.cfg (in the home directory)
4. /etc/ansible/ansible.cfg

Lection #5. Best practices

Ansible config

```
[defaults]
inventory              = hosts.ini
callback_whitelist     = profile_tasks
host_key_checking      = False
display_skipped_hosts  = False
display_args_to_stdout = True
stdout_callback        = debug
remote_user            = deploy
roles_path             = ./roles
library                = ./library
vault_password_file    = ~/ansible-vault.pass
private_key_file       = /path/to/file


[ssh_connection]
pipelining             = True
ssh_args               = -o ControlMaster=auto -o ControlPersist=600s

```

Lection #6. Usecases

Ansible vault

The vault feature can encrypt any structured data file used by Ansible.

```
1  echo "Creating Encrypted Files"
2  ansible-vault create ~/ansible-vault.pass
3
4  echo "Editing Encrypted Files"
5  ansible-vault edit ~/ansible-vault.pass
6
7  echo "Running a Playbook with Vault"
8  ansible-playbook site.yml –ask-vault-pass
9  ansible-playbook site.yml –vault-password-file ~/ansible-vault.pass
10
11 cat > ansible.cfg << EOL
12 [defaults]
13 vault_password_file   = ~/ansible-vault.pass
14 EOL
```

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

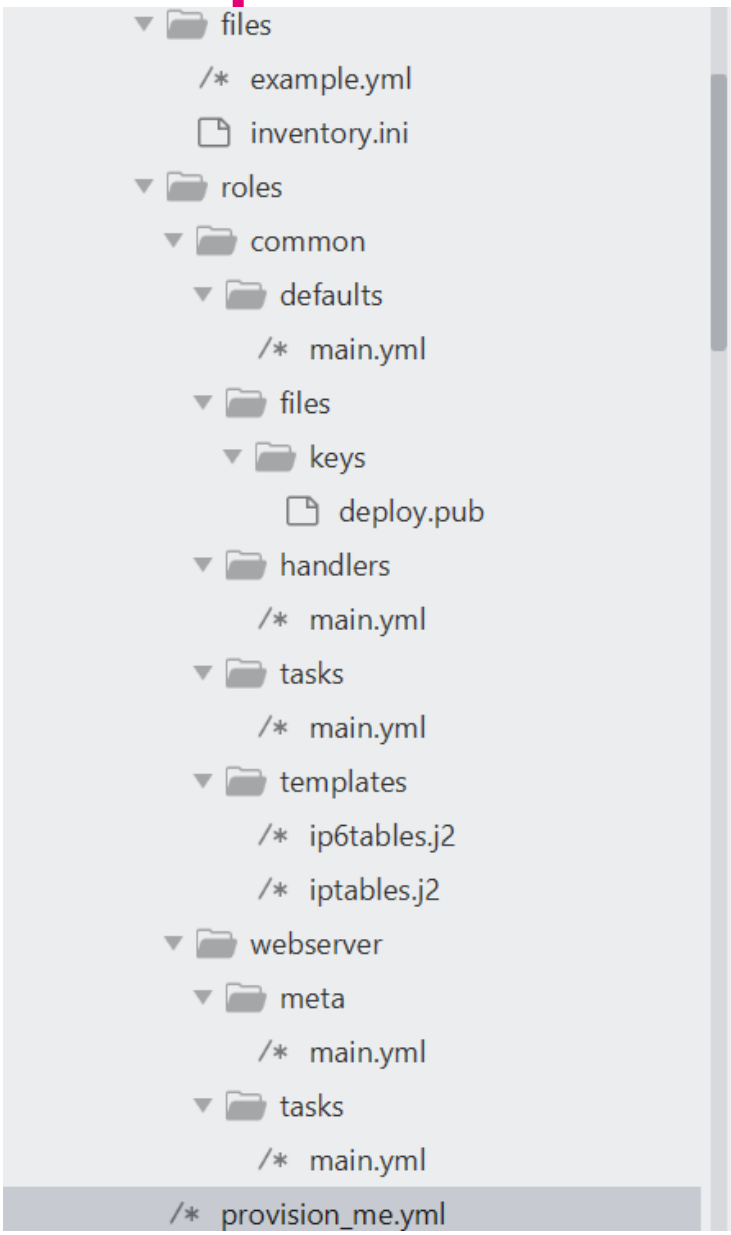
Lection #5. Best practices
Roles

A bundle of related tasks/handlers/templates

- Defaults - default variables for the role
- Files - contains files which can be deployed via this role.
- Handlers - contains handlers, which may be used by this role or even anywhere outside this role.
- Meta - defines some meta data for this role
- Tasks - contains the main list of tasks to be executed by the role
- Templates - contains templates which can be deployed via this role
- Vars - other variables for the role

Lection #5. Best practices

Roles




```yaml
- name: provision server
  hosts: all
  become: True
  become_user: root
  vars:
    iptables_allowed_ports:
      - {protocol: tcp, port: 80}
      - {protocol: tcp, port: 443}
      - {protocol: udp, port: 161}
  roles:
    - webserver
  tasks:
    - name: copy files
      copy:
        src: "{{ item }}"
        dest: "/home/vagrant/{{ item }}"
      with_items:
        - inventory.ini
        - example.yml
```

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 #5. Best practices

Inventory.

1. Lists all hosts which Ansible may manage
2. Simple "INI" format or YAML
3. Can defne groups of hosts
4. Default is /etc/ansible/hosts
   - We will instead use ./hosts.local
   - Can override using -i <filename>

Lection #5. Best practices

Inventory. Variables

1. You can set variables on hosts or groups of hosts
2. Variables can make tasks behave differently when applied to different hosts
3. Variables can be inserted into templates
4. Some variables control how Ansible connects

Lection #5. Best practices

Inventory. Static

```yaml
---
azure:
  hosts:
    azureenv01:
      ansible_port: 443
      ansible_host: 51.4.16.19
      ansible_user: admin01
    azureenv02:
      ansible_port: 443
      ansible_host: itrain02.germanycentral.cloudapp.microsoftazure.de
      ansible_user: admin01
    azureenv03:
      ansible_port: 443
      ansible_host: train03.germanycentral.cloudapp.microsoftazure.de
      ansible_user: admin01
  vars:
    http_proxy: http://127.0.0.1:4444
    https_proxy: http://127.0.0.1:4444
    no_proxy: 127.0.0.1
```

Lection #5. Best practices

Inventory. Static

```
[azure]
imagemastertrain01
imagemastertrain02 ansible_host=1.1.1.1 ansible_port=443 ansible_user=admin01
imagemastertrain03 ansible_host=imagemastertrain03.germanycentral.cloudapp.microsoftazure.de

[azure:vars]
http_proxy=http://127.0.0.1:4444
https_proxy=http://127.0.0.1:4444
no_proxy=127.0.0.1
```

Lection #5. Best practices

Inventory. Host_vars

1. Directly in the inventory (hosts) fle
2. In file host_vars/imagemastertrain01

```
http_proxy: http://127.0.0.1:4444
https_proxy: http://127.0.0.1:4444
no_proxy: 127.0.0.1
some_strange_var:
  - foo
  - bar
```

Lection #5. Best practices
Inventory. Group_vars

- group_vars/azure

```
http_proxy: http://127.0.0.1:4444
https_proxy: http://127.0.0.1:4444
no_proxy: 127.0.0.1
some_strange_var:
  - foo
  - bar
```

- group_vars/all

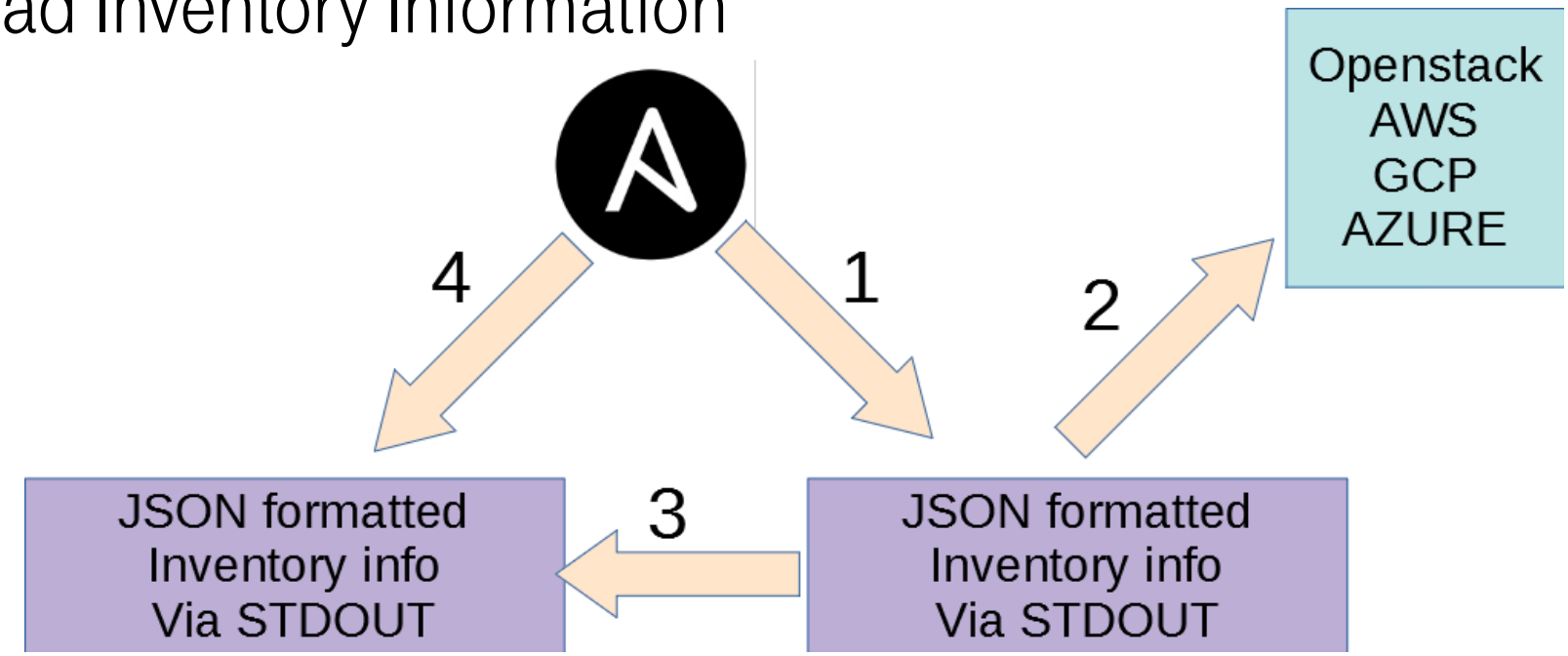
More YAML, applies to every host

Note: host vars take priority over group vars

Lection #5. Best practices

Inventory. Dynamic

1. Execute Dynamic Inventory
2. Collect Target information
3. Output Inventory to STDOUT
4. Read Inventory Information



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 #5. Best practices

Repository structure

```
production                # inventory file for production servers
staging                   # inventory file for staging environment

group_vars/
   group1.yml             # here we assign variables to particular groups
   group2.yml
host_vars/
   hostname1.yml          # here we assign variables to particular systems
   hostname2.yml

library/                  # if any custom modules, put them here (optional)
module_utils/             # if any custom module_utils to support modules, put them here (optional)
filter_plugins/           # if any custom filter plugins, put them here (optional)

site.yml                  # master playbook
webservers.yml            # playbook for webserver tier
dbservers.yml             # playbook for dbserver tier

roles/
    common/               # this hierarchy represents a "role"
        tasks/            #
            main.yml      #  <-- tasks file can include smaller files if warranted
        handlers/         #
            main.yml      #  <-- handlers file
        templates/        #  <-- files for use with the template resource
            ntp.conf.j2   #  <------- templates end in .j2
        files/            #
            bar.txt       #  <-- files for use with the copy resource
            foo.sh        #  <-- script files for use with the script resource
        vars/             #
            main.yml      #  <-- variables associated with this role
        defaults/         #
            main.yml      #  <-- default lower priority variables for this role
        meta/             #
            main.yml      #  <-- role dependencies
        library/          # roles can also include custom modules
        module_utils/     # roles can also include custom module_utils
        lookup_plugins/   # or other types of plugins, like lookup in this case

    webtier/              # same kind of structure as "common" was above, done for the webtier role
    monitoring/           # ""
    fooapp/               # ""
```

Lection #5. Best practices

Repository structure

```
inventories/
   production/
      hosts               # inventory file for production servers
      group_vars/
         group1.yml       # here we assign variables to particular groups
         group2.yml
      host_vars/
         hostname1.yml    # here we assign variables to particular systems
         hostname2.yml

   staging/
      hosts               # inventory file for staging environment
      group_vars/
         group1.yml       # here we assign variables to particular groups
         group2.yml
      host_vars/
         stagehost1.yml   # here we assign variables to particular systems
         stagehost2.yml

library/
module_utils/
filter_plugins/

site.yml
webservers.yml
dbservers.yml

roles/
    common/
    webtier/
    monitoring/
    fooapp/
```

Lection #5. Best practices

- Use dynamic inventory with clouds
- Differentiate staging vs production
- Roles
- Ansible doesn't reduce count of strings, it allows you to reduce complexity of modifying your infrastructure.

https://docs.ansible.com/ansible/latest/user_guide/playbooks_best_practices.html

Lection #5. Best practices
Workshop

```
1 $env:http_proxy='http://spbsrv-proxy2.t-systems.ru:3128'
2 $env:https_proxy='http://spbsrv-proxy2.t-systems.ru:3128'
3 git clone http://projects.t-systems.ru/lgonchar/ansible-course-public.git
4 cd student_files/05
5 vagrant up –provider hyperv
```

Lection #5. Best practices
Homework

- Create new role snmpd – it should
  - Install snmpd
  - Configure snmpd
  - Get snmp community string from ansible-vault
  - Get snmp community string as parameter for role
- Modify existing webserver role
  - Generate self signed cert
  - Configure HTTPS

THANK YOU!

Q&A

> Use the ansible, Luke
>
> Obi Wan Kenobi

lev.goncharov@t-systems.com