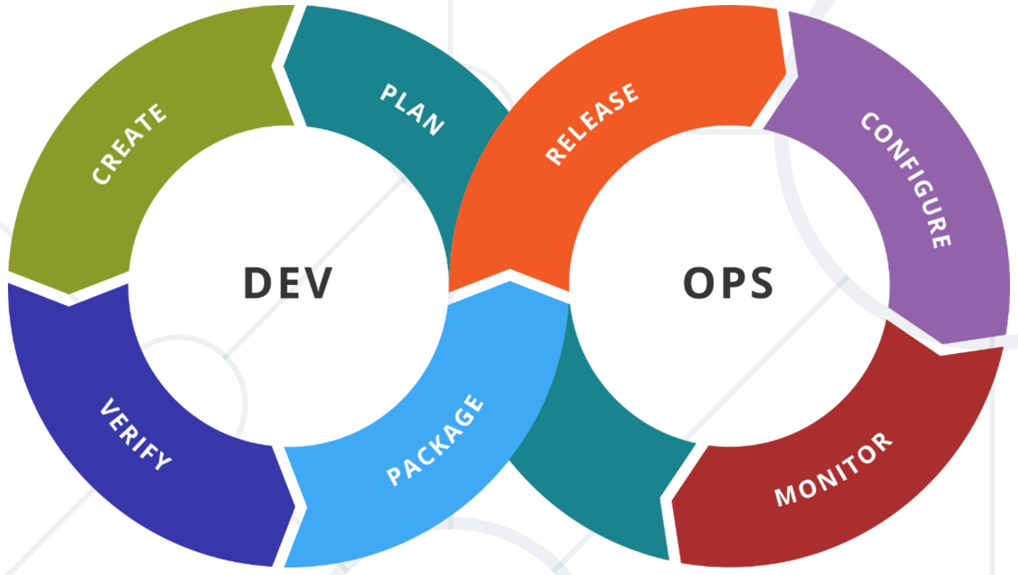
Ansible

Introduction and Basic Techniques



SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://softuni.org

You Have Questions?

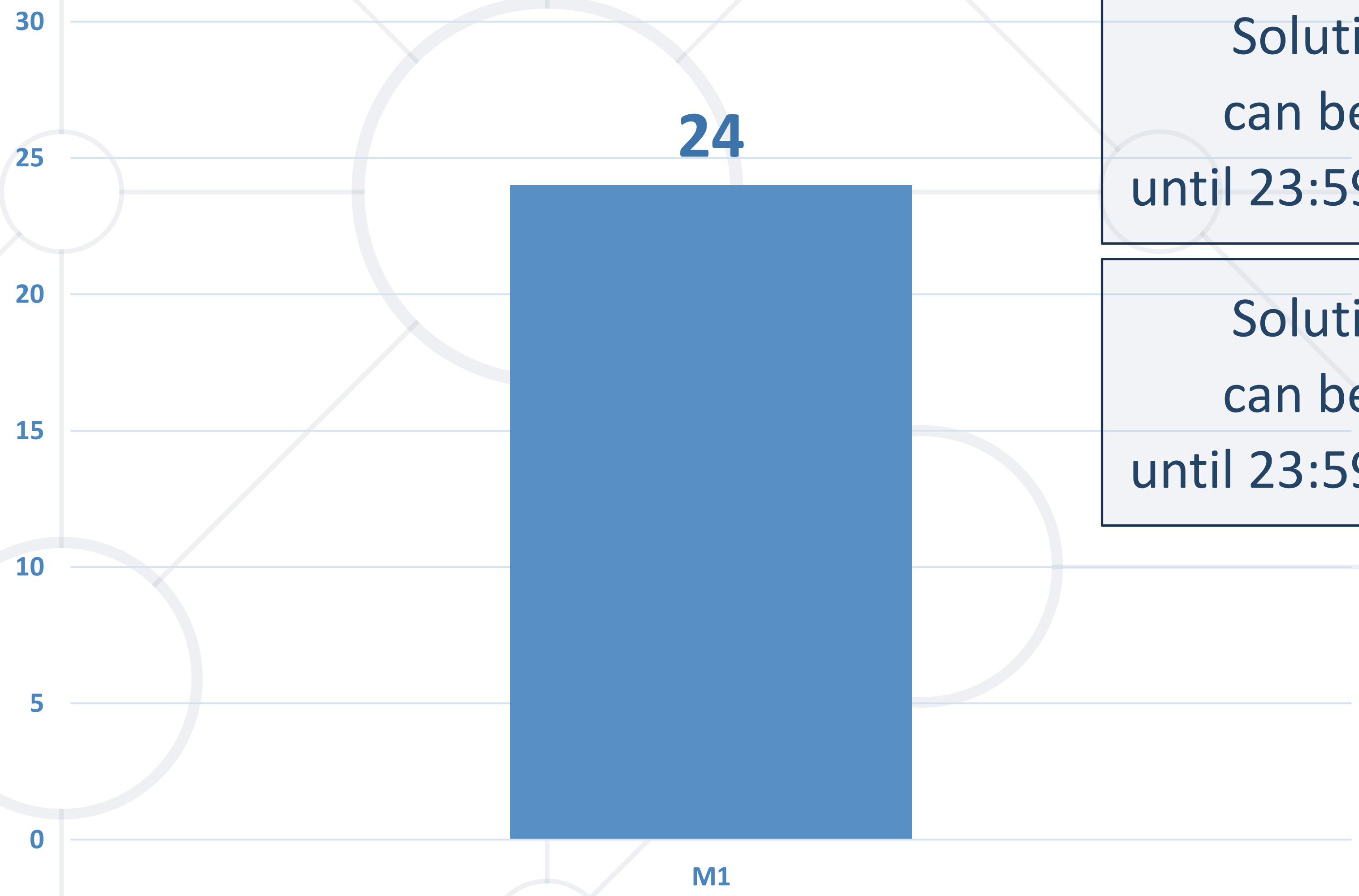
sli.do

#DevOps-23

facebook.com/groups

/DevOpsInfrastructureandConfigManagementApril2023

Homework Progress



Solutions for M1
can be submitted
until 23:59 on 12.04.2023

Solutions for M2
can be submitted
until 23:59 on 19.04.2023

Previous Module (M1)

Quick Overview

What We Covered

1. Infrastructure as Code
   - Introduction
   - Terraform Basics
2. Terraform and Docker
3. Terraform and AWS

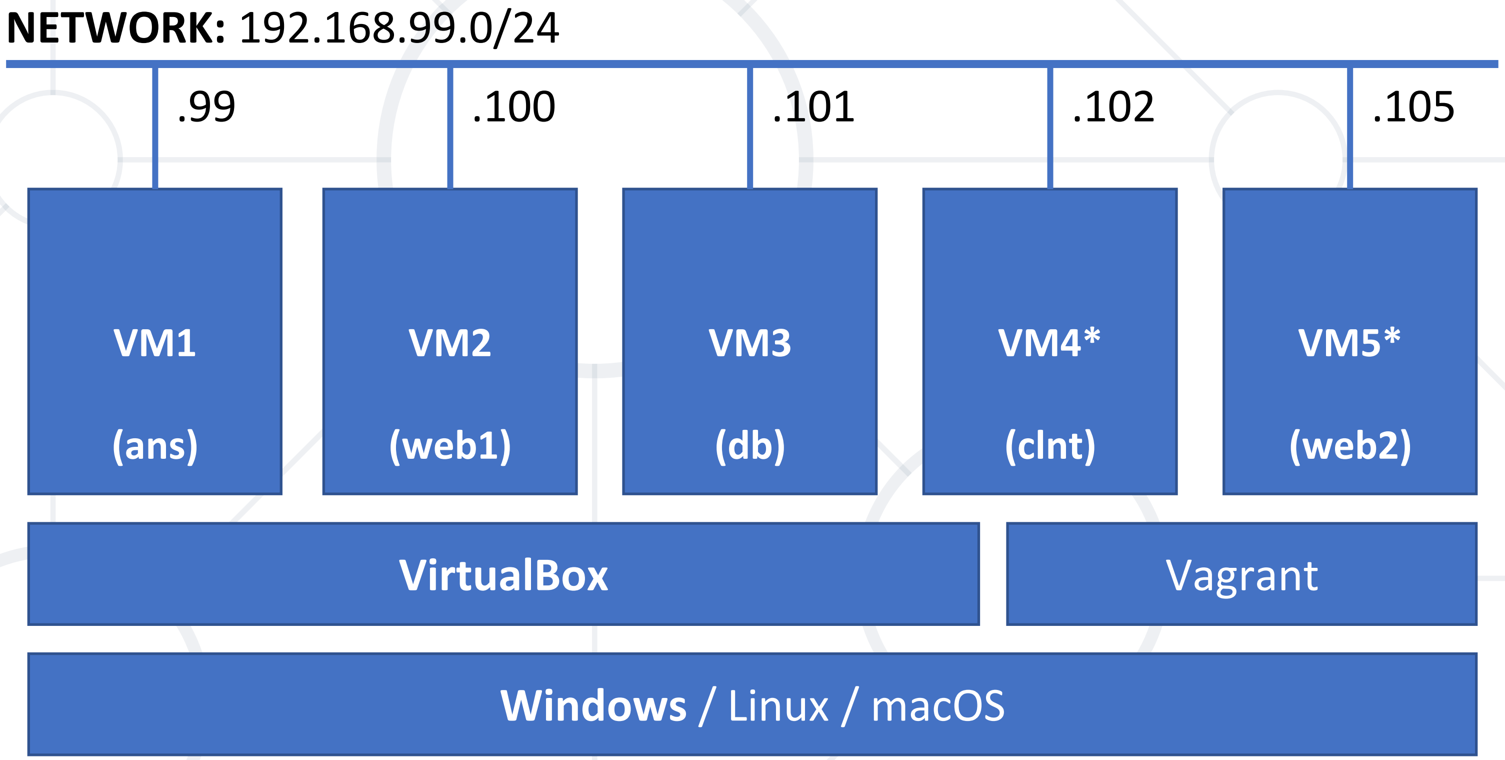
This Module (M2)

Topics and Lab Infrastructure

Table of Contents

1. Introduction to Ansible
   - Other solutions
   - Ansible architecture
2. Working with Ansible
   - Work with Inventories and Configurations
   - Using Modules
3. Advanced Ansible
   - Playbooks and Roles

Lab Infrastructure



* VM4 and VM5 can be skipped. Of course, the exercises should be adjusted accordingly

DevOps

Available Solutions

For Provisioning, Configuration, and etc.

The Need

- Manage efficiently large-scale infrastructures
- Replicated environments
- Avoid the so-called Snowflake servers
- Version control for the environment
- Quick provisioning
- Quick recovery

Solutions (1)

- **Chef** by Chef (now Progress)
  - Recipes are written in Ruby DSL
  - Master-agent model, pull-based approach
  - Supports Windows both as server and node
- **Puppet** by Puppet
  - Recipes are written in Ruby DSL and Embedded Ruby
  - Master-agent model
  - Supports agents on Linux, OS X, and Windows

Solutions (2)

- **Salt** by SaltStack
  - Recipes are written in YAML
  - Two modes – with or without agents (Salt Minions)
  - Supports Windows both as host and remote system
- **Ansible** by Ansible Inc (now Red Hat)
  - Recipes are written in YAML
  - Agentless
  - Windows is only supported as remote system

Introduction to Ansible

Architecture. Components. Installation

What is Ansible*?

"… An **ansible** is a category of fictional device or technology capable of instantaneous or faster-than-light communication…"

* https://en.wikipedia.org/wiki/Ansible

What is/does Ansible?

- **Change Management**
  - Define and track system state. Idempotence
- **Provisioning**
  - Transition ***form a State A to a State B***
- **Automation**
  - Automatic execution of tasks on a system
- **Orchestration**
  - Coordination of automation between systems

Key Characteristics

- No extra components, just the bare minimum
  - There are no agents, repositories, etc.
- Easy to learn and program
  - Uses YAML, structured, easy to read and write
- Secure by design
  - Uses ***OpenSSH*** and ***WinRM***, ***root*** and ***sudo***
- Open and extendable
  - Shell commands, Library (Ansible-Galaxy) with tons of modules

Requirements

- Ansible Control Server
  - Python 2.7+ / 3.5+
  - Linux/Unix/Mac
  - Windows is not supported
- Remote Server
  - Linux/Unix/Mac – Python 2.6+ / 3.5+, SSH
  - Windows – Remote PowerShell

Current version
2.9.xx
(Red Hat release)

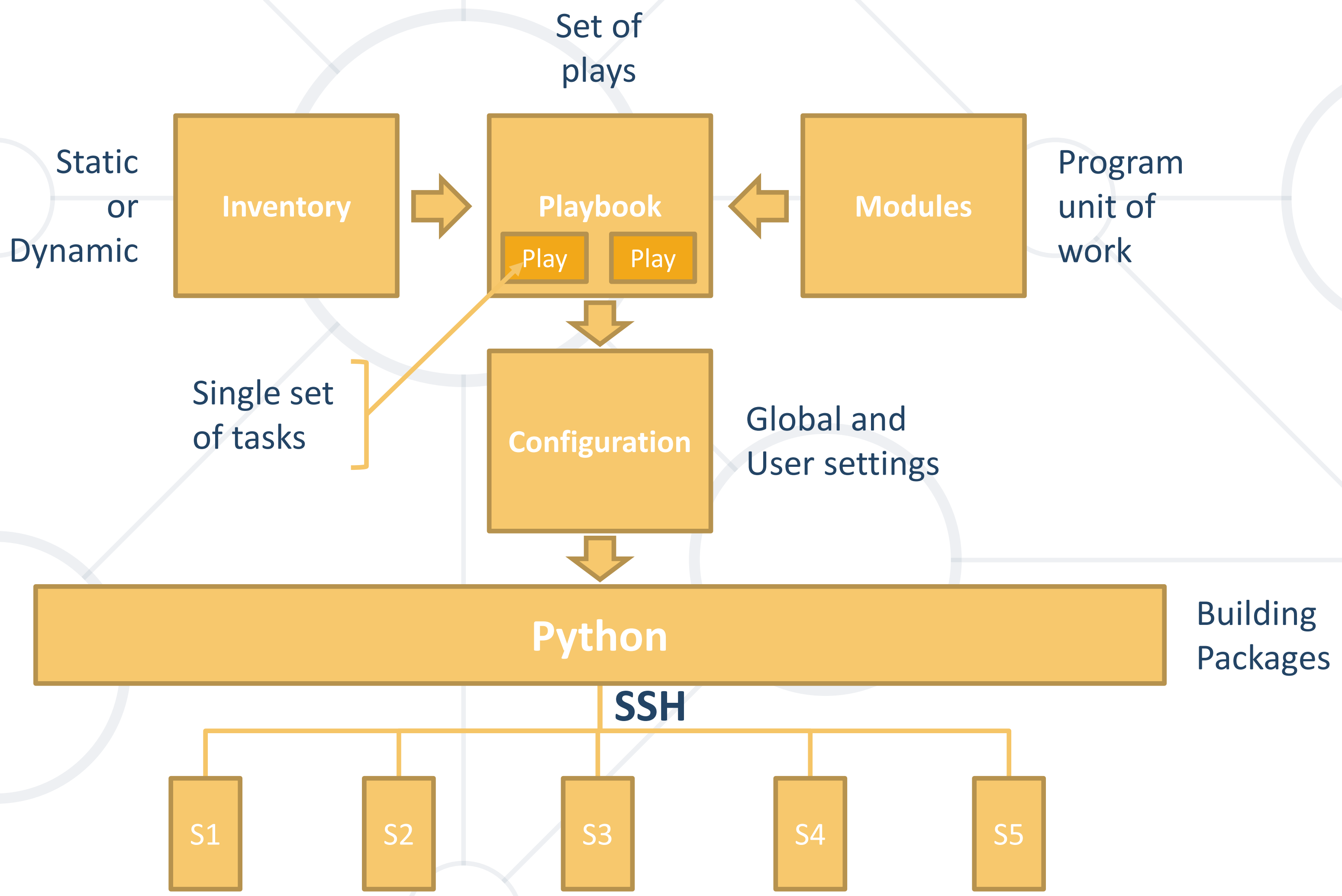
Red Hat Release vs Community Release

- Starting from version 2.10 we have two artifacts
- Community package called **ansible** (current version is **7.x**)
    - If contains the Ansible language and runtime + range of community curated collections
    - It is based and expands on what was included in Ansible 2.9
- Minimalist package called **ansible-core** (current version is **2.14.x**)
    - In version 2.10 was called **ansible-base**
    - It contains the Ansible language, runtime and a short list of core modules and plugins
- Both can be installed via OS **package manager** or with **pip**

Availability

- Compilation from source
- Installation from the official repositories
  - Supports all major distributions
  - Usually, additional repository have to be added
    - RedHat 6.x – **EPEL** / RedHat 7.x – **Extras**
    - SUSE Enterprise Linux 12.x/15.x – **Package Hub** Repository
    - Older versions of Ubuntu – **Ansible PPA** (ppa:ansible/ansible)
- Installation via pip (Python package manager)

Architecture and Components



Practice: Installation. Environment Setup

Live Demonstration in Class

Inventory

Manage Your Hosts

Inventory

- Define and describe the environment
  - Reflect our interpretation of the environment
- Can be stored anywhere on the system
  - Locally for a project, user, etc.
- Can have more than one inventory file
  - We can choose at run-time or use a configuration file

Inventory Features

- Behavioral Parameters
- Groups
- Groups of Groups
- Assign Variable
- Scale out
- Either Static or Dynamic

Two default groups – **all** and **ungrouped**

Sample Inventory File

```
web ansible_host=192.168.82.100
clnt ansible_host=192.168.82.102 ansible_user=vagrant ansible_ssh_pass=vagrant
```

Behavioral Parameters

```
[servers]
web

[stations]
clnt
```

Groups

```
[machines:children]
servers
stations
```

Groups of Groups

```
[machines:vars]
ansible_user=vagrant
ansible_ssh_pass=vagrant
```

Variables

Inventory Two Ways

- The **INI Way**

```
host.dob.lab

[web]
w1.dob.lab
w2.dob.lab

[db]
db1.dob.lab
```

- The **YAML Way**

```
all:
  hosts:
    host.dob.lab:
  children:
    web:
      hosts:
        w1.dob.lab:
        w2.dob.lab:
    db:
      hosts:
        db1.dob.lab:
```

Scale Out

- Split the inventory file
  - On smaller more manageable pieces
  - Chose a criteria – location, environment, role
  - Store the files in the same directory – shared variables
  - Store the files in separate directories
- Once split the files it is difficult to merge them

Variable Precedence and Files

- Order of precedence
  - Group Variables (**group_vars**) All
  - Group Variables (**group_vars**) GroupName
  - Host Variables (**host_vars**) HostName
- Variable Files

YAML file indication

Comment

Variables

```
---
# file: host_vars/web01
username: user01
userdir: /home/user01
```

DevOps

Configuration

Variables and Settings

Configuration Storage and Order

- Configuration Files
  - **$ANSIBLE_CONFIG**
  - **./ansible.cfg**
  - **~/.ansible.cfg**
  - **/etc/ansible/ansible.cfg** ⇨ Not created if built from source

- They are **not merged**, the **first found** is taken into account
- **Override** by prefixing the name with **$ANSIBLE_<setting>**

DevOps

Modules

Modules

- Modules do the **actual work**
- They can be executed
  - Manually using the **ansible** command
  - In batches with **ansible-playbook**

- They are known as **task plugins** or **library plugins**
- Two major types – **Core** and **Extras**
- Organized in categories – **Command**, **Files**, **System**, etc.

Modules Help

- List all available modules

```
$ ansible-doc -l
```

- Get detailed information for a module

```
$ ansible-doc service
```

- Show playbook snippet for a module

```
$ ansible-doc -s service
```

DevOps

Practice: See It in Action

Live Demonstration in Class

Plays and Playbooks

Fundamentals

Plays

- Plays map hosts to tasks
- Each play can have multiple tasks
- Tasks call modules
- Tasks run sequentially

Play Declaration

```
- hosts: webservers
  become: true
```

Global Play Declaration

```
  tasks:
    - name: Copy new index.html
      copy: src=html/index.html dest=/var/www/html/
```

Module

Task Declaration

Playbooks

- A playbook contain **one** or **more plays**
- Stored in **YAML files**
- Two ways of declaration – **list** and **dictionary**
- Can be used to **build** entire **application environment**

Playbook File

```
---
- hosts: webservers
  become: true
  tasks:
    - name: Install Apache HTTP Server
      dnf: name=httpd state=present
    - name: Start Apache HTTP Server and Enable it
      service: name=httpd state=started enabled=true
```

Play One

```
- hosts: databases
  become: true
  tasks:
    - name: Install MariaDB Server
      dnf: name=mariadb,mariadb-server state=present
    - name: Start and enable MariaDB
      service: name=mariadb state=started enabled=true
```

Play Two

Playbook Two Ways

- The **List Way**

```
---
- hosts: web
  become: true
  tasks:
    - name: Install WEB
      dnf: name=httpd state=present
    - name: Start WEB
      service: name=httpd state=started
```

- The **Dictionary Way**

```
---
- hosts: web
  become: true
  tasks:
    - name: Install WEB
      dnf:
        name: httpd
        state: present
    - name: Start WEB
      service:
        name: httpd
        state: started
```

Playbooks Execution

- Execute with default inventory

```
$ ansible-playbook playbook_name.yml
```

- Execute with specified inventory

```
$ ansible-playbook -i inventory playbook_name.yml
```

- On host failure it is excluded from further tasks execution
- Failed hosts are stored in a file
- Retry execution only for failed hosts

```
$ ansible-playbook book.yml --limit @/path/to/file
```

DevOps

Roles

Fundamentals

Roles

- Allow **easy sharing** of content
- Way of automatic loading of tasks, vars, and handlers
- Described via **YAML files** in certain directory structure
- Search for roles
  - A **roles/** directory relative to the playbook file
  - By default, in **/etc/ansible/roles**

Roles Directory Structure

- **tasks** – main list of tasks to be executed
- **handlers** – handlers, that may be executed
- **defaults** – default variables for the role
- **vars** – other variables for the role
- **files** – files, that can be deployed
- **templates** – templates, that can be deployed
- **meta** – meta data (parameters and dependencies)

* **main.yml** is expected in each folder
** **Other** task specific **files** can be **included**, like ***redhat.yml***
*** **At least one** of the folders must be included

Role Example

- Definition (main.yml)

```
---
  - name: Firewall | Open HTTP port
    firewalld:
      service: http
      permanent: true
      state: enabled
      immediate: true
```

- Usage (playbook.yml)

```
---
- hosts: web
  roles:
    - firewall-8080
```

```
.
├── ansible.cfg
├── hosts
├── playbook.yml
└── roles
    ├── firewall-8080
    │   └── tasks
    │       └── main.yml
    └── firewall-http
        └── tasks
            └── main.yml
```

Ansible Galaxy

- Free site for **finding**, **downloading**, and **sharing** roles
- We can **develop** and **share** our own roles. **GitHub account** is needed
- Galaxy can be run **on-premise** as well
- Command line tool **ansible-galaxy** is included

```
$ ansible-galaxy install username.role
```

- Default storage is configured via **roles_path** variable
- Install a role to a **custom path**

```
$ ansible-galaxy install --roles-path . username.role
```

- Install roles included in requirements file

```
$ ansible-galaxy install -r requirements.yml
```

DevOps

Additional Techniques

Include Files

- Easier playbook management – smaller playbooks
- Reuse other playbooks – common/repeatable plays
- Can load external variable

```
tasks:
  - include_vars: ext_var_file.yml
  - include: web-server.yml
  - include: db-server.yml
```

Register Task

- Link tasks – data from one task is passed to another
- Can be used for error catching

```
tasks:
  - shell: /usr/bin/whoami
    register: username
  - file: path=/path/to/folder/readme.txt
          owner={{ username }}
```

Debug Module

- Display output during execution
- Easier problem identification
- Two ways for execution

```
tasks:
  - debug: msg="Host: {{ inventory_hostname }}"

  - shell: /usr/bin/uptime
    register: result
  - debug:
      var: result
      verbosity: 2
```

Playbook Handlers

- Runs when notified
- It is notified only when state=changed
- Runs last

```
tasks:
  - copy: src=files/httpd.conf dest=/etc/httpd/conf/
    notify:
      - Web Server Restart

handlers:
  - name: Web Server Restart
    service: name=apache2 state=restarted
```

Conditional Clause - When

- Evaluate should a task execute

```
tasks:
  - apt: name=apache2 state=present
    when: ansible_os_family == "Debian"

  - dnf: name=httpd state=present
    when: ansible_os_family == "RedHat"
```

- Use APT module if Debian or use DNF/YUM if RedHat

Conditional Clause - Result

- Track execution status of the previous task
- Status options – **success**, **failed**, **skipped**
- Should add **ignore_errors** or the playbook will fail

```
tasks:
  - command: /bin/false
    register: result
    ignore_errors: True

  - command: /bin/some_command
    when: result|failed
```

Templates

- Jinja2 Engine
- Create and copy dynamic files

```
- name: Deploy index.j2 on RedHat
  vars:
    v_host_type: RedHat
  template: src=templates/index.j2
            dest=/var/www/html/index.html
  when: ansible_os_family == "RedHat"
```

- **templates/index.j2**

```
<h2>Hello from Ansible on {{ v_host_type }}!</h2>
```

Practice: Playbooks in Action

Live Demonstration in Class

Summary

- Ansible is a powerful solution for **configuration** and **provisio ning**
  - It can be installed from **source**, **repository**, or **PIP**
  - It is driven by a set of configuration files
- **One** or **more inventories** can be used simultaneously
- Actual executable parts are called **modules**
- Modules can be combined in **plays**
- Plays can be combined in **playbooks**
- Plays can go one step further with **Jinja2 templates**

Resources

- Ansible Documentation
  http://docs.ansible.com/
- Ansible Modules
  http://docs.ansible.com/ansible/latest/list_of_all_modules.html
- Ansible Galaxy
  https://galaxy.ansible.com/
- Ansible Galaxy Documentation
  https://galaxy.ansible.com/docs/
- Ansible Examples Repository
  https://github.com/ansible/ansible-examples
- Short Ansible Tutorial
  https://www.codereviewvideos.com/course/ansible-tutorial

Software University

Questions?

SoftUni

Software University

SoftUni Svetlina

SoftUni Creative

SoftUni Digital

SoftUni Foundation

SoftUni Kids

SoftUni Diamond Partners

Software University

SCHWARZ

LIDL

Kaufland

SCHWARZ Produktion

pre zero

Coca-Cola HBC Bulgaria

POKERSTARS | CAREERS

Postbank

Решения за твоето утре

AMBITIONED

DXC TECHNOLOGY

SOFTWARE GROUP

Bosch.IO

INDEAVR

Serving the high achievers

DRAFT KINGS

PHAR VISION

SmartIT

createX

SUPER HOSTING .BG

Educational Partners

CODE IT UP

License

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Pr ofession and Job for Software Developers
  - softuni.bg, softuni.org
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg