Introduction to Ansible

Keeping configurations nice and accurate

steffann

About the presenter

- Sander Steffann
  - Independent IPv6 consultant
  - Strategy, architecture, design, labs, testing, implementation etc…
- Involvement
  - Global NOG Alliance
  - RIPE Community
  - RIPE NCC Executive Board

- Introduction
- Automation in general
  - Benefits of automation
  - What to automate?
  - What not (yet) to automate?
  - Available automation tools (Ansible, Puppet, Salt etc)

- Working with Ansible
  - What is Ansible?
  - Installing Ansible
  - How Ansible Works and its Key Components
  - Using the Ad-Hoc ansible command

- Ansible playbooks
  - YAML syntax
  - Creating an inventory
  - Playbook Basics
  - Available Ansible modules
  - Organising playbooks into roles

Benefits of automation

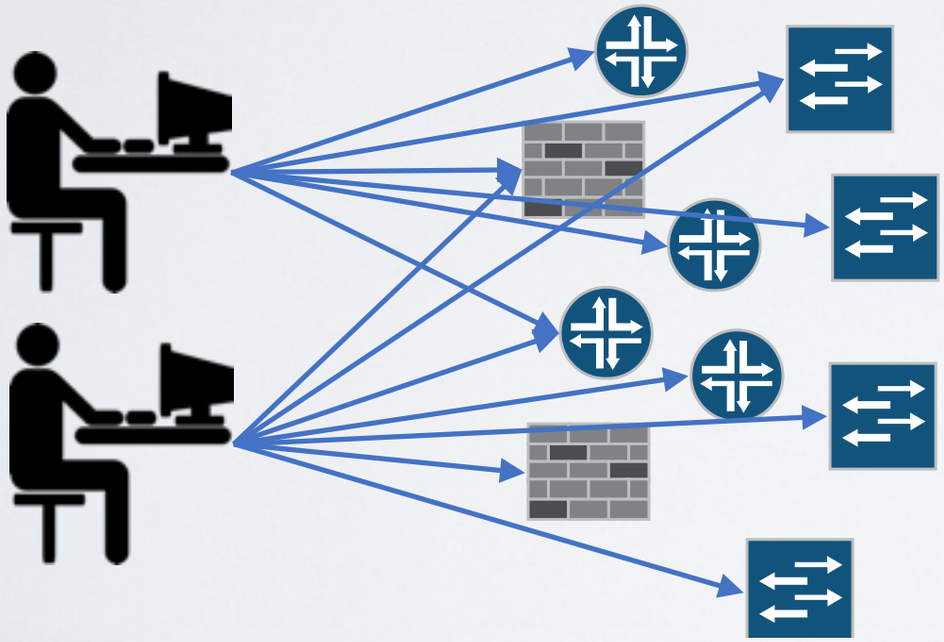
steffann

COMMON PROBLEMS

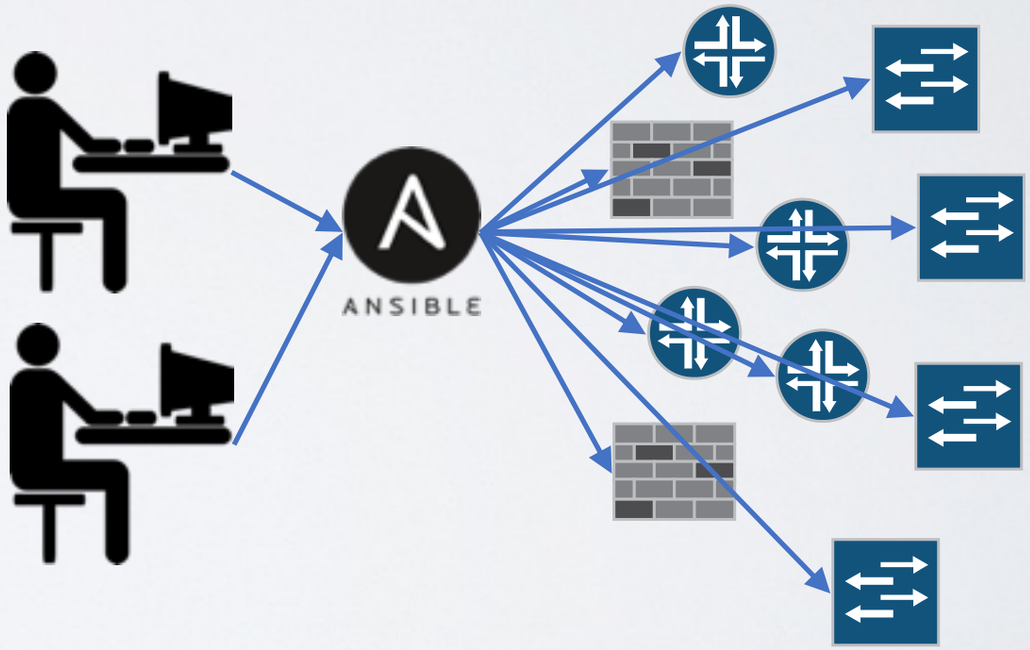
steffann

Rough comparison

Traditional



Automated



What to automate?

Or not?

Or not yet?

Extreme viewpoints

- Automate everything!
  - Every server, router, switch, firewall etc. should be under complete automation control!
  - All or nothing approach
  - Possible for Greenfield deployment
  - Good automation skills need to be already available

Extreme viewpoints

- Use automation only for daily tasks
  - Still configure devices manually
  - Create automation for operational changes
    - Update software
    - Create new user
    - Add BGP neighbour
    - Etc…

The middle ground

- Look at your devices
  - What changes a lot?
  - What do you spend most time on?
  - Where are most errors made?
  - What would you like to do, but is too time-consuming?
- Start with automating those

Automation tools

steffann

Deployment models

- Push: controller pushes config to device
  - Can be agent-less
  - *Ansible, Otter, SaltStack, Terraform*
- Pull: device pulls config from controller
  - Requires agent on device
  - *Ansible-pull, CFEngine, Chef, Otter, Puppet, SaltStack*

Configuration style

- Declarative
  - Describes “what”
  - Focus on desired end-state, automation makes it happen
  - *Ansible, CFEngine, Otter, Puppet, SaltStack, Terraform*
- Imperative
  - Describes “how”
  - Focus on step-by-step actions, automation executes it
  - *Ansible, Chef, Otter, SaltStack*

Why Ansible?

- Open source
- Constantly improved
- Easily extensible (if you know Python)
- Can manage many network devices
- De-facto standard by now

What is Ansible?

What is Ansible?

- Configuration management
  - Consistency
  - Central administration
    - What state should the device configuration be in?
- Provisioning
  - Take steps to make sure the state is correct

Philosophy

- Ansible is not a programming language
  - Describe "what" you want, not "how" (mostly)
- Keep things simple and understandable
  - Think of it more as a modelling system (like Lego?) than a programming language
  - Advanced users can write plugins to do complex work
  - The playbook should be simple to read and check

How does that work?

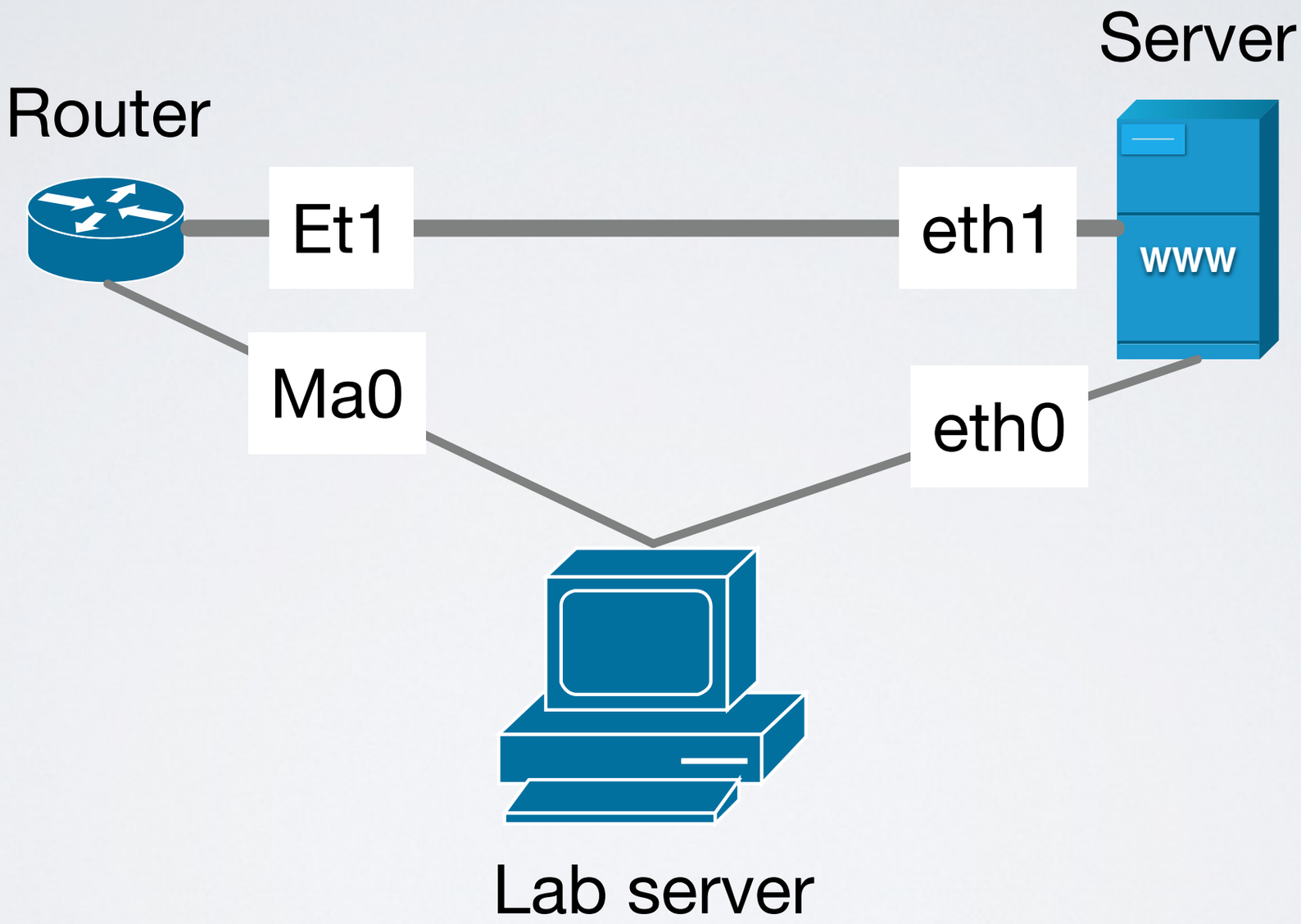
- Configuration
  - Define sets of configuration items, one for each purpose
  - Define which devices belong to which set
  - Add device-specific settings to that
- Result
  - You end up with consistently configured devices
  - Where each device's purpose is clearly defined

Lab introduction

Time Constraints

- We don't have time to do the labs
  - I will demo each exercise for the classroom
- You can do the exercises in your own time
  - The lab will remain online until the beginning of June
  - Or run it on your own laptop!
- Support
  - If you have any questions feel free to approach me!
  - I can often be found in the NomCom room

TOPOLOGY



Lab Introduction

- We use netlab (https://netlab.tools/)
  - Tool to quickly set up network labs for testing and training
  - Command-line-only, no web interface or remote desktop
- Student login:
  - Protocol: SSH
  - Server: ripe90.lab.steffann.nl
  - Username: student1, student2, student3, …, student26
  - Password: «on your Terminal Access Card»

STARTING A LAB

ssh ⌥⌘2

13% 24 GB 43 kB↓ 296 kB↑ 87 x 23

```
teacher1@ripe90:~$ cd exercises/ansible/
teacher1@ripe90:~/exercises/ansible$ ls
router.j2  server.j2  topology.yml
teacher1@ripe90:~/exercises/ansible$ cat topology.yml
provider: libvirt

nodes:
  router:
    provider: clab
    device: eos
    config: router.j2
  server:
    device: linux
    config: server.j2
    memory: 512

links:
  - router-server
teacher1@ripe90:~/exercises/ansible$ netlab up
```

Device logins

- User student1:
  - ssh admin@lab1-router
  - ssh admin@lab1-server
- User student2:
  - ssh admin@lab2-router
  - ssh admin@lab2-server
- Etc…

How Ansible Works and its Key Components

At a high level

- Connectivity
  - Use SSH (+ e.g. netconf) to connect to devices
- Method
  - Upload Python scripts to execute on device
- Execute actions
  - Gather information (Facts)
  - Apply configuration items based on definitions and facts

Components

- Inventory
  - The list of devices, optionally organised in groups
- Host and Group variables
  - Every device and group can have its own settings
- Facts
  - Automatically discovered variables about a device

Components

- Modules
  - Action types that you can use in your automation
- Task
  - An action to perform
- Roles
  - A set of tasks

Components

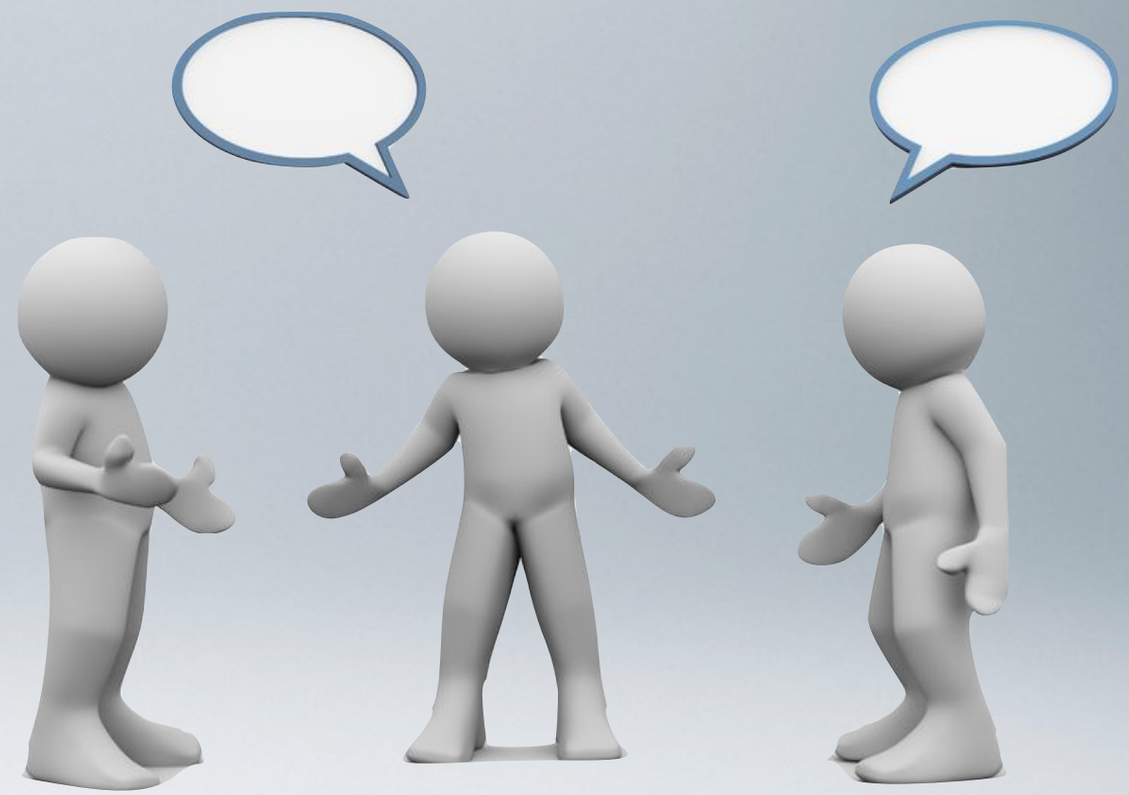
- Play
  - A set of hosts/groups connected to a set of tasks
- Playbook
  - A list of plays

Components

- Tags:
  - A label attached to a task
- Filters:
  - Filter, extract and/or convert data from variables
- Lookup:
  - Lookup data from external service
- Tests:
  - Check whether a variable matches a certain condition

Discussion

Get a clear picture of the structure



Using the Ad-Hoc ANSIBLE COMMAND

Is the lab reachable?

- Let's make sure we can connect:

```
ssh admin@labXX-server
ssh admin@labXX-router
```

- Save the fingerprint to known_hosts
  - Ansible needs it to be present
  - So let's use this opportunity to add it

A SIMPLE START

- We first need a directory and an inventory:

```
mkdir ~/ansible && cd ~/ansible
echo "labXX-server" > hosts
```

- Let's try:

```
ansible --inventory hosts
   --module-name command
   --args "lsb_release -d"
   labXX-server
```

- What happened?

A SIMPLE START

- Ansible needs to be able to log in!
- Let's try:

```
ansible --inventory hosts
   --module-name command
   --args "lsb_release -d"
   --user admin
   --ask-pass
   labXX-server
```

- What happened?

A SIMPLE START

- Ansible needs to be able to log in!
- Let's try:

```
ansible --inventory hosts
   --module-name command
   --args "id"
   --user admin
   --ask-pass
   labXX-server
```

- What happened?

A SIMPLE START

- Now again with root permission:

```
ansible --inventory hosts
   --module-name command
   --args "id"
   --user admin
   --ask-pass
   --become
   --ask-become-pass
   labXX-server
```

- Success!

Discussion

What do you think so far?

steffann

Creating an inventory

steffann

Permissions

- Ansible uses /etc/ansible by default
  - We want to work in ~/ansible
- Create a basic ansible.cfg in ~/ansible
- vi/emacs/nano/joe ~/ansible/ansible.cfg

```
[defaults]
inventory = ./hosts
interpreter_python = auto_silent
```

CREATE A REAL INVENTORY

- We just configured the inventory as:

  `./hosts`

- Let's fill it!

```
[servers]
labXX-server

[routers]
labXX-router
```

Check

- It should now look like this:

```
$ ls -l ~/ansible
-rw-r--r-- 1 lab lab   31 Sep 26 21:40 ansible.cfg
-rw-r--r-- 1 lab lab   42 Sep 26 21:41 hosts

$ cat ~/ansible/ansible.cfg
[defaults]
inventory = ./hosts
interpreter_python = auto_silent
```

Check

- It should now look like this:

```
$ cat ~/ansible/hosts
[servers]
labXX-server

[routers]
labXX-router
```

Groups

- We now have three groups:
  - servers
  - routers
  - all

Let's use it

- Let's try the ansible command:

```
cd ~/ansible
ansible
--module-name command
--args "lsb_release -d"
--user admin
--ask-pass
servers
```

- Success!

TYPING PASSWORD IS ANNOYING

- Create the following file:

```
~/ansible/group_vars/all.yml
```

- And put this in it:

```
ansible_user: "admin"
ansible_ssh_pass: "admin"
ansible_become_pass: "admin"
```

- Now ansible can authenticate itself

CHECK

- It should now look like this:

```
$ ls -l ~/ansible
-rw-r--r-- 1 lab lab   31 Sep 26 21:40 ansible.cfg
drwxr-xr-x 2 lab lab 4096 Sep 26 21:17 group_vars
-rw-r--r-- 1 lab lab   42 Sep 26 21:41 hosts

$ cat ~/ansible/group_vars/all.yml
ansible_user: "admin"
ansible_ssh_pass: "admin"
ansible_become_pass: "admin"
```

Look at the facts

- Let's try:

```
cd ~/ansible
ansible
--module-name setup
servers
```

- Success!

Look at the facts again

- Let's try:

```
cd ~/ansible
ansible -m setup servers
```

- Success!

Discussion

Improvement compared to previous steps

Playbook Basics

steffann

Components

- Play
  - A set of hosts/groups connected to a set of tasks
- Playbook
  - A list of plays

First we need some config

- Create the following file:

  `~/ansible/group_vars/routers.yml`

- And put this in it:

```
ansible_connection: network_cli
ansible_become_method: enable
ansible_network_os: eos
```

DEFINED IN YAML

- An example, ~/ansible/versions.yml:

```
- hosts: servers
  tasks:
  - debug: msg="{{ansible_facts.lsb.release}}"

- hosts: routers
  tasks:
  - debug:
      msg: "{{ansible_net_version}}"
```

RUNNING THE PLAYBOOK

- We can execute the playbook with:

```
cd ~/ansible
ansible-playbook versions.yml
```

Discussion

Now we are getting somewhere

steffann

ORGANISING PLAYBOOKS INTO ROLES

steffann

Reusing common tasks

- Some tasks will be used often
  - Basic sysadmin settings
  - Installing or updating software
  - Etc.

Using roles

- What are roles?
  - Roles are collections of tasks, files, templates etc.
- Common structure

```
~/ansible/roles/role_name/
  defaults/
  files/
  handlers/
  tasks/
  templates/
```

Default settings

Files to copy to device

Tasks that run automatically

The main tasks of the role

Templates for text files

A SIMPLE ROLE

- Let's start with roles for getting versions
  - In ~/ansible/roles/server_version/tasks/main.yml

```
- name: "Get server version"
  debug: msg="{{ansible_facts.lsb.codename}}"
```

  - In ~/ansible/roles/router_version/tasks/main.yml

```
- name: "Show router version"
  debug:
    msg: "Version {{ansible_net_version}}"
```

How to use roles

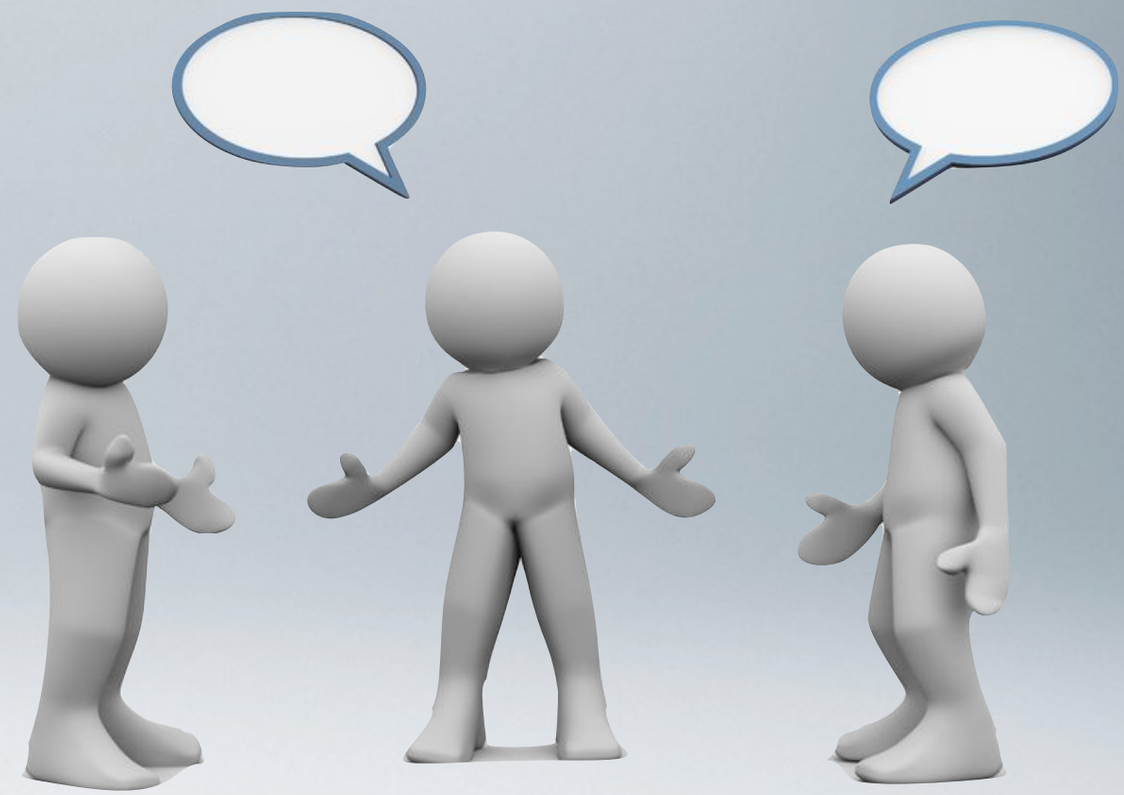
- From your playbook

```
- hosts: servers
  roles:
    - server_version

- hosts: routers
  roles:
  - router_version
```

Discussion

What would be good roles?



steffann

SECURING YOUR DATA WITH ANSIBLE-VAULT

steffann

Confidential information

- Your inventory contains very secret info
  - Usernames and password
  - Encryption keys?
  - Etc.
- What if this information leaks?

Encrypt data with vault

- Encrypting whole files
  - The simplest way to use it

```
ansible-vault encrypt group_vars/all.yml
ansible-vault edit group_vars/all.yml
```

- You can also encrypt `files/` in a role!

ENCRYPT DATA WITH VAULT

- Encrypting single settings
  - For encrypting parts of a file

```
ansible-vault encrypt_string "This is a secret"
```

  - Use the output in your group_vars, host_vars etc.

```
the_secret: !vault |
        $ANSIBLE_VAULT;1.1;AES256
        62313365396662343061393464336163383764376134333665396363534333632666535333761
        6339626533396638616637363262653932616635
```

Letting ansible decrypt

- Ansible will still understand vault data
  - Use `--ask-vault-pass` to supply the password
  - You can make that the default in ansible.cfg

```
ask_vault_pass = True
```

Available Ansible modules

Modules

- Modules provide functionality
- This is where Ansible is very strong
  - Modules for Linux, BSD, Windows etc.
  - Modules for network devices
  - Modules for VMWare, AWS, Kubernetes etc.
  - Etc.
  - Etc..
  - Etc…

A BRIEF OVERVIEW

- Just some of the most interesting ones
  - According to me personally: this list is biased!
  - Look around on the Ansible website to see for yourself

Ansible Utilities

- assert
- debug
- fail
- Import
  - import_playbook
  - import_role
  - import_tasks
- Include
  - include_role
  - include_tasks
  - include_vars
- pause
- set_fact
- wait_for

Generic commands

- command
- expect
- raw
- script
- shell

Files

- archive
- unarchive
- blockinfile
- lineinfile
- fetch
- file
- ini_file
- patch
- replace
- stat
- synchronize
- tempfile
- template
- xml

Networking

- Arista
- Aruba
- Cisco
  - ASA
  - IOS / -XR
  - Nexus
  - WLC
- Cumulus
- Dell
  - os6
  - os9
  - os10
- A10
- F5
- Fortinet
- Huawei
- Juniper
- Mikrotik
- Ubiquiti

OS Packages

- apk
- apt
- dnf
- dpkg
- flatpak
- homebrew
- opkg
- pkg5
- pkgng
- ports
- rhn
- rpm
- yum
- zypper

Source control

- Bazaar
- Bitbucket
- Git
  - GitHub
  - GitLab
- Mercurial (hg)
- Subversion (svn)

System

- Filesystem
  - iSCSI
  - LVM
  - Parted
- Cron / at
- SSH
  - authorized_key
  - known_hosts
- Hostname
- Firewall
  - IP Tables
  - Firewalld
  - UFW
- Make
- Ping
- SELinux
- Systemd
- Timezone
- User / group