Introducing Ansible's "network-engine" Role

Nick Russo
NETWORK ENGINEER

@nickrusso42518   www.njrusmc.net

Agenda

Why should I care?

Reviewing "ansible-galaxy"

Refactoring with "command_parser"

Parsing best practices

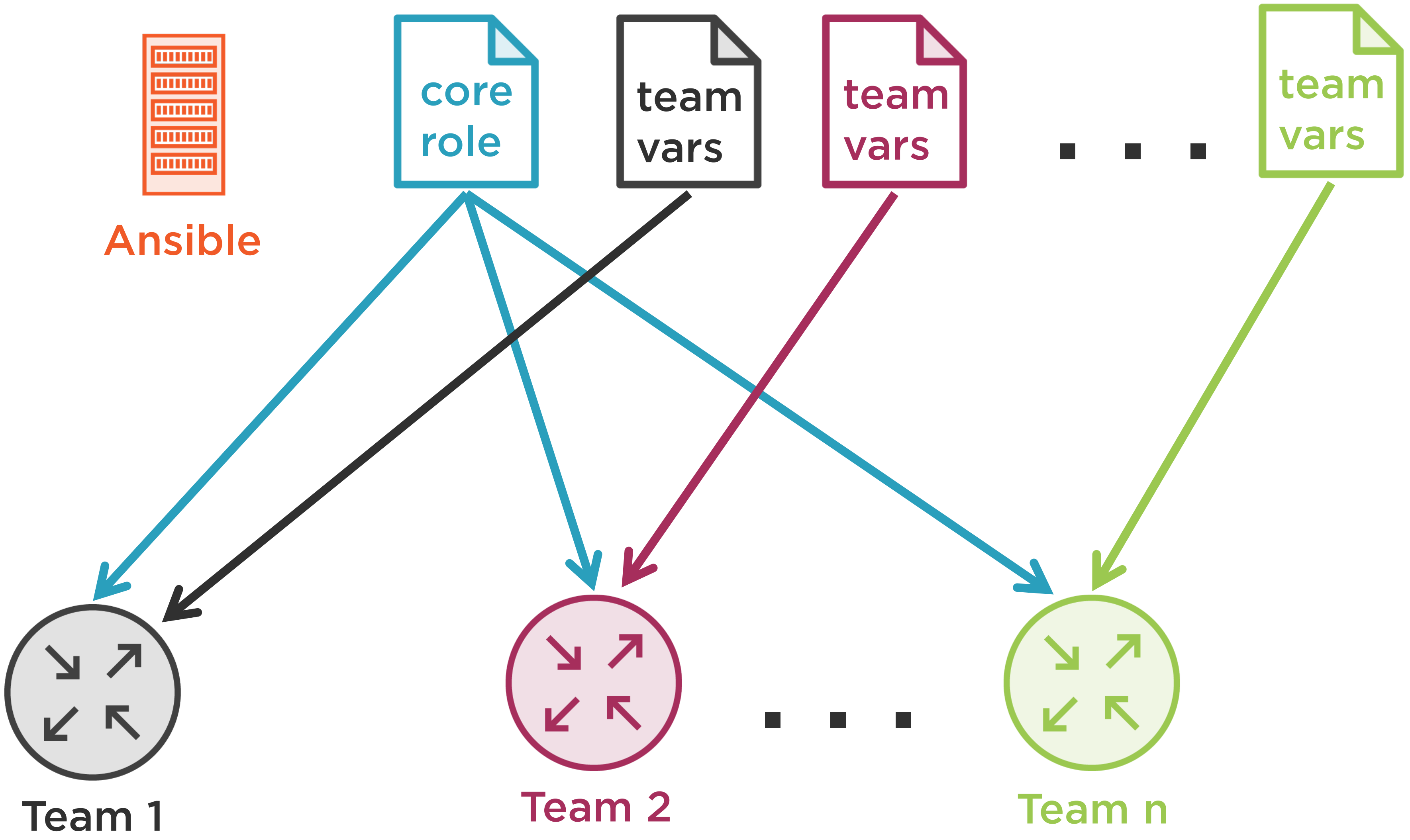
Why Do Roles Matter?

Portability

Built-in tests

Scalability

Use Case: Scaling with Roles



Understanding "ansible-galaxy"

Build and install roles

The role marketplace

galaxy.ansible.com

Demo

Working with ansible-galaxy

```
-   pattern_match: "vrf def..."
  register: vrf_defs

- pattern_group:
   - pattern_match: VRF name
   - pattern_match: RT import
   - pattern_match: RT export
  loop through vrf_defs

-   json_template
    (do proper formatting)
  export JSON as dict
```

◂ Find the "vrf definition" chunks

◂ Grab VRF name

◂ Grab VRF import route-targets

◂ Grab VRF export route-targets

◂ For each "vrf definition" chunk

◂ Return data in correct format

```
- name: "Use role to parse"
  hosts: routers
  roles:
    - "ansible-network.
         network-engine"
  tasks:
   - name: "Parse RT data"
      command_parser:
        file: vrf.yml
        content: "{{ text }}"
```

◀ Specify roles to include (line split for readability only)

◀ Don't call custom filter to parse, instead use "command_parser"

Demo

Writing a command parser

Demo

Integrating the parser into the playbook

Advantages of Each Technique

Custom Python parser	Ansible command_parser
More flexible	Simpler (usually)
Less overall code	Easier to learn
Best for complex/non-standard text	Best for repetitive/predictable text
Portable into other Python frameworks	Officially supported by Ansible