Scaling Network Operations with Modular Ansible

Joseph Nicholson
Network Operations Engineer
Global IP Network - NTT DATA

NANOG™

NTT DATA

Introduction

NANOG™

Who is Joseph Nicholson?*

Network Operations Engineer

- 26 Years Experience as a Network Engineer

18 Years with NTT DATA

- 12 Years on the NOC team
- 7 Years on Network Operations and Network Scaling team focusing on Network Automation

NANOG™

*not an existential question

Who is NTT DATA?

Global IP Network (GIN) Division

- Tier 1 Transit Provider - AS2914
- Network spans 5 continents, 40+ countries, 90+ PoPs

Services Offered

- IP Transit, Virtual Link, Global Virtual Link, DDOS Protection
- 1G, 10G, 100G, 400G port speeds

800+ Network Elements

- Routers – 250+ Backbone (BB) across 2 vendors
- DWDM – 100+ across 2 vendors (transponders and line systems)

Disclaimer!

This is the way that worked for us

- *If it ain't broke…*

Change is hard

- This can be improved greatly but change can be hard the deeper you get down the rabbit hole.

Task File Size

- I know someone is going to say my task files are too long

Ansible Usage

When do we use Ansible?

- Router processes based on existing method of procedure (MoP)

Why am I using via a repository and on the CLI?

- It was easier for everyone in the beginning
- Every user has a copy in their local home directory

AWX transition started

- Slowly migrating older playbooks to AWX
- New playbooks live here from day 1

Ansible Repository

NANOG™

Ansible Playbook Overview

Ran manually on CLI	Ran through AWX
Router Software Upgrades	Vendor TAC Collection
Update Traffic	Software Package Management
Circuit Testing	Maintenance Snapshots
Router Post Turnup Checks	Optical Software Upgrades*

Git Repository

Operational Ansible Playbooks

- Have a repository naming policy!

Gitlab hosted repository

CI/CD Pipeline

Custom runner

- Secured to prevent unauthorized usage

Git Repository

Branches

- main
- awx_branch
- awx_ee_build
- awx_inventory
- awx_management

Main Branch Root Folder Structure

Main Folder

- Playbooks run here

Focusing on…

- Main branch tasks

```
.
├── collections
├── inventory
├── roles
├── scripts
├── tasks
├── vars
├── circuit_test.py
├── circuit_testing.yaml
├── maint_snaps.yaml
├── post_turnup_check.yaml
├── router_sw_mgmt.yaml
├── router_upgrade.yaml
└── update_traffic.yaml
```

Main Branch Sub-folder Structure

- Sub-folders are named after the playbooks
- Divided into NOS based folders
- ”Shared” tasks are used by all playbooks
  - non-router based

```
tasks
├── circuit_testing
├── collection
├── maint_snaps
├── post_turnup
├── router_upgrade
│   ├── iosxr
│   ├── junos
│   └── sros
├── shared
├── sw_mgmt
└── update_traffic
```

AWX Branch Root Folder Structure

Playbooks run as roles

Tasks Folder

- Task files are shared between multiple roles

Playbook folder

- Mostly legacy in this branch

```
.
├── collections
├── group_vars
├── molecule
├── playbooks
├── roles
├── scripts
├── tasks
└── vars
```

AWX Branch Sub-folder Structure

- Folder retains same structure
- Most task files are contained within the roles folder

CI/CD Pipeline

main

- FIX ME
  - Finds #FIXME tags
  - Stops processing MR
- AWX Updates
  - Syncs os_revision vars file to AWX

awx_branch

- ansible and python linting and syntax checks

CI/CD Pipeline

awx_ee_build

- Generates new images as test images
- Can promote test images to production
- Can revert old images back to production

awx_inventory

- Builds AWX inventory files and installs them using AWX API

awx_management

- No pipeline

Modular Tasks

NANOG™

Task Files

What are task files?

- Task files are collection of tasks to perform a process

Keep them on task

- It's ok you can groan here

Task across playbooks

- Task files for one playbook can be used by other playbooks

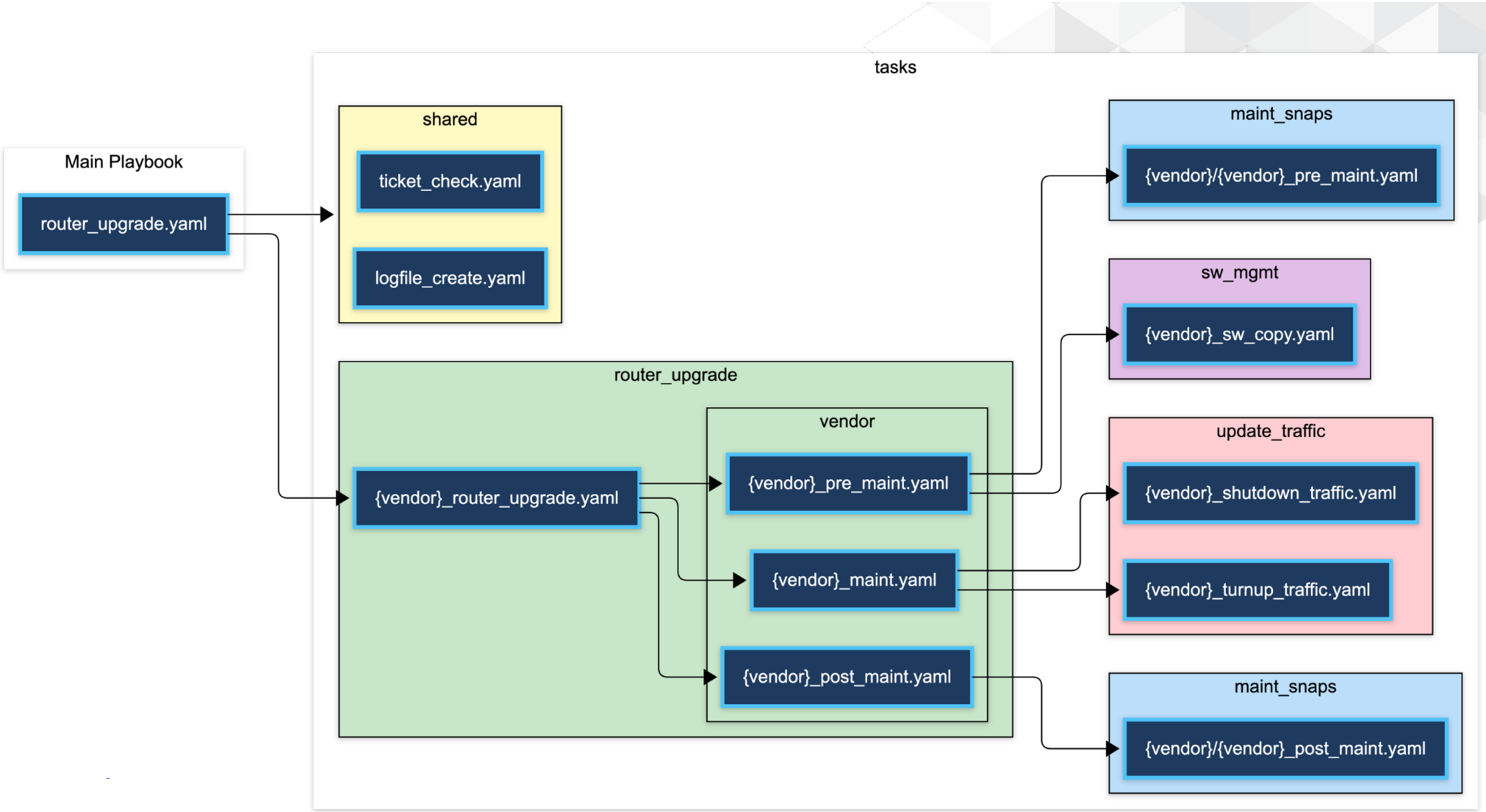
When task files are called, they run their tasks.

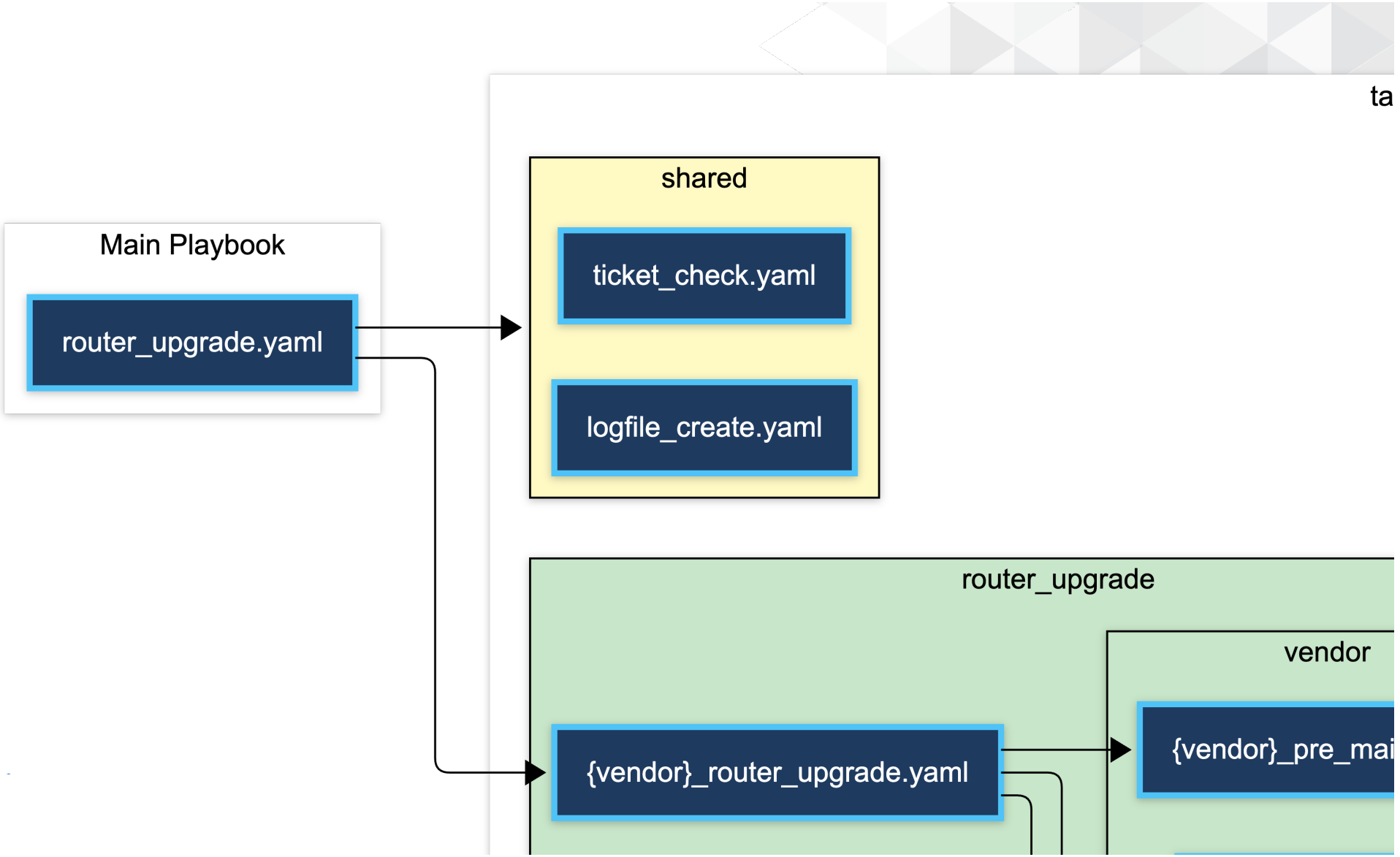
- Don’t need to be redirected back to the original playbook

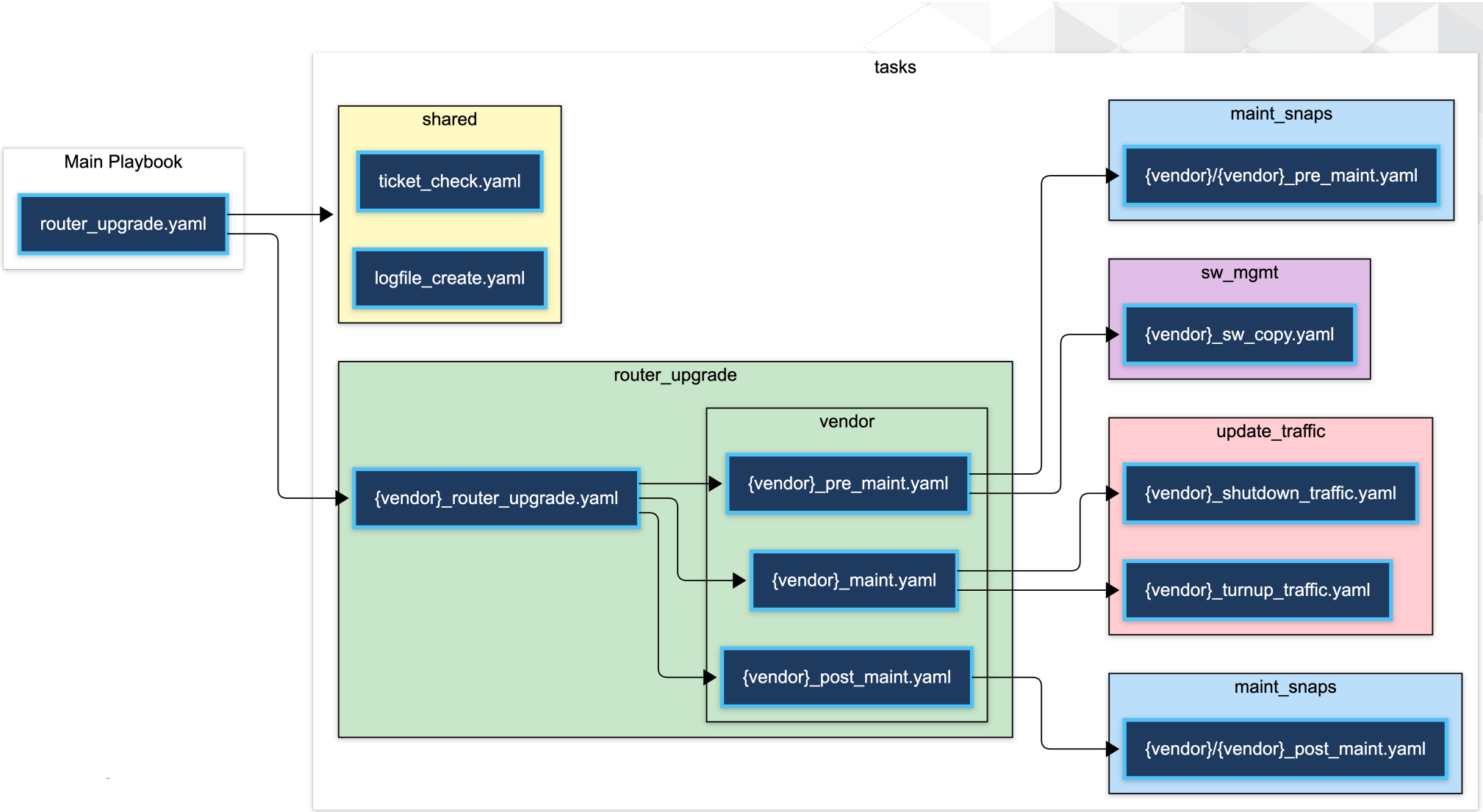
Router Software Upgrade Example

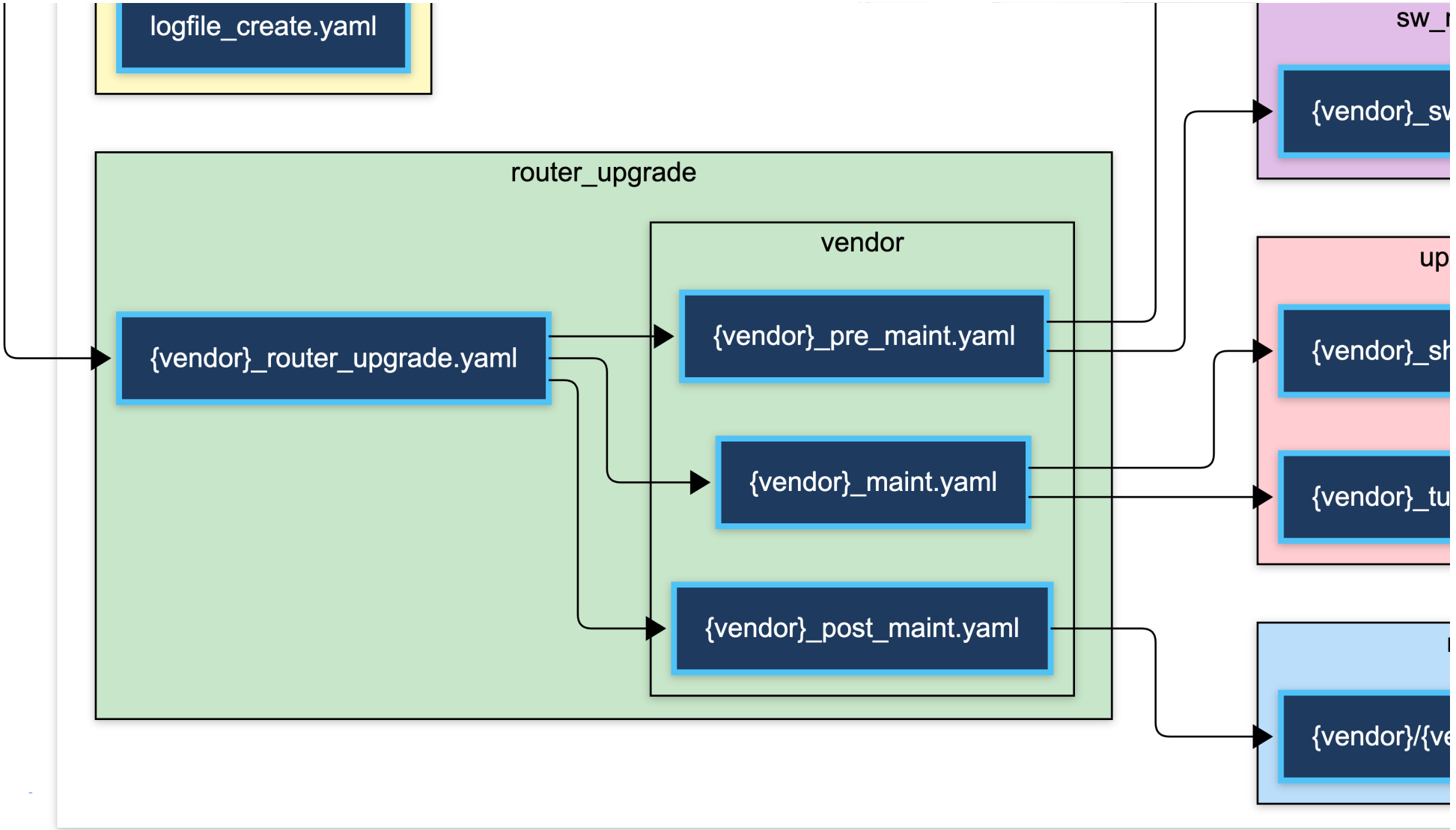
- Software upgrades
- Software management
- Maintenance Snapshots
- Traffic Updates

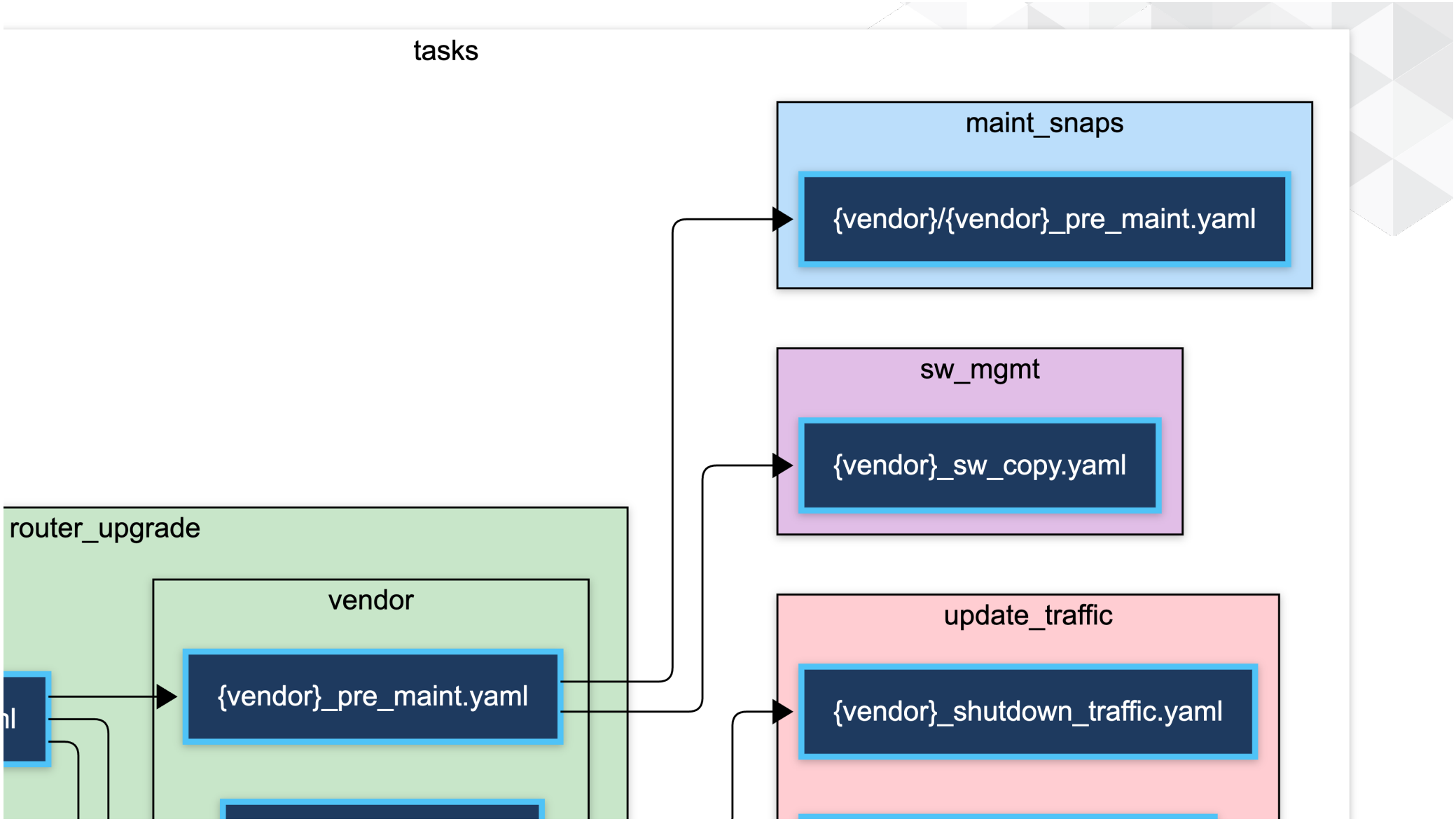
Not a through step-by-step account of every task in the procedure

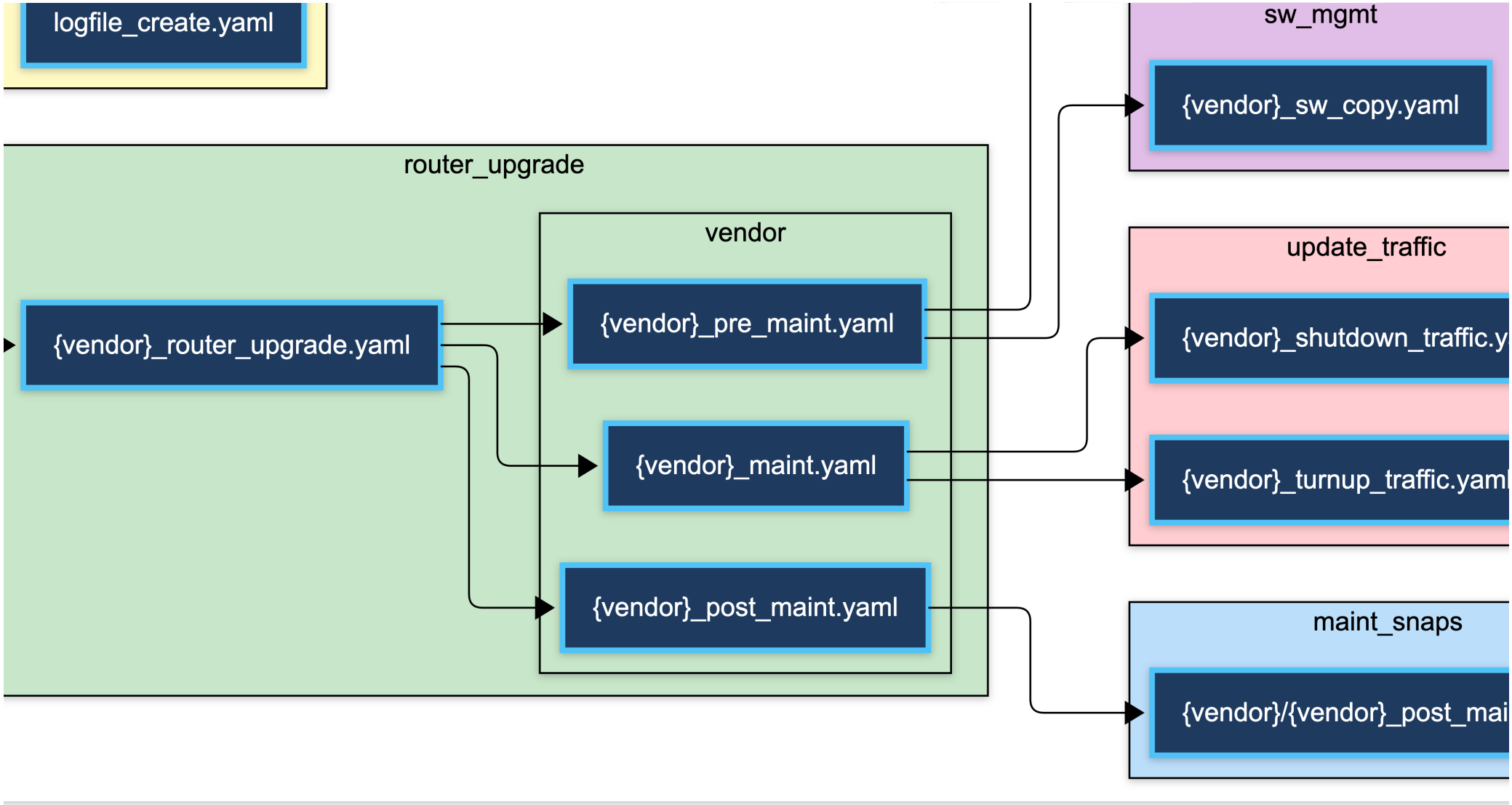


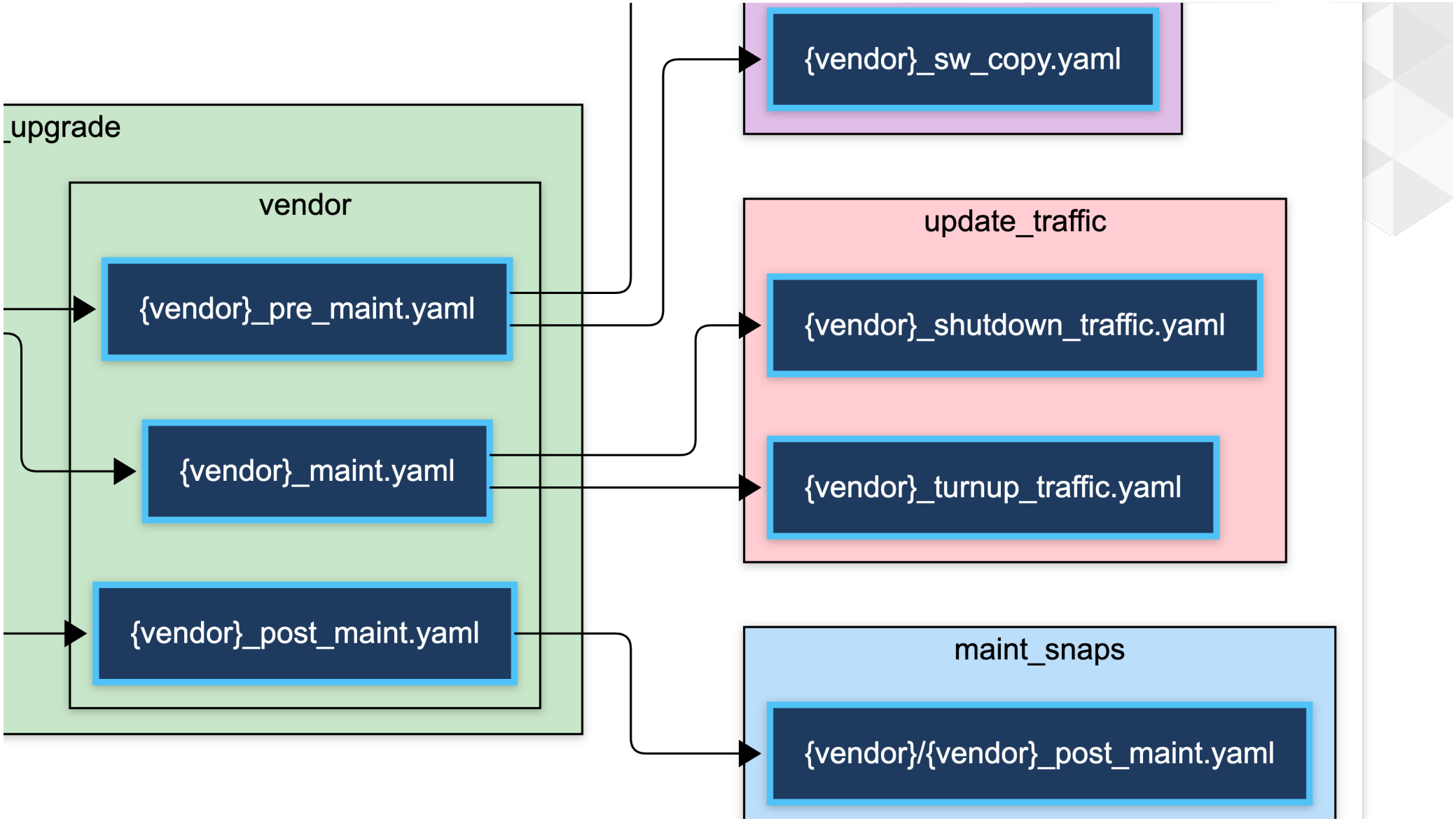

ta
shared
ticket_check.yaml
logfile_create.yaml
Main Playbook
router_upgrade.yaml
router_upgrade
vendor
{vendor}_router_upgrade.yaml
{vendor}_pre_mai

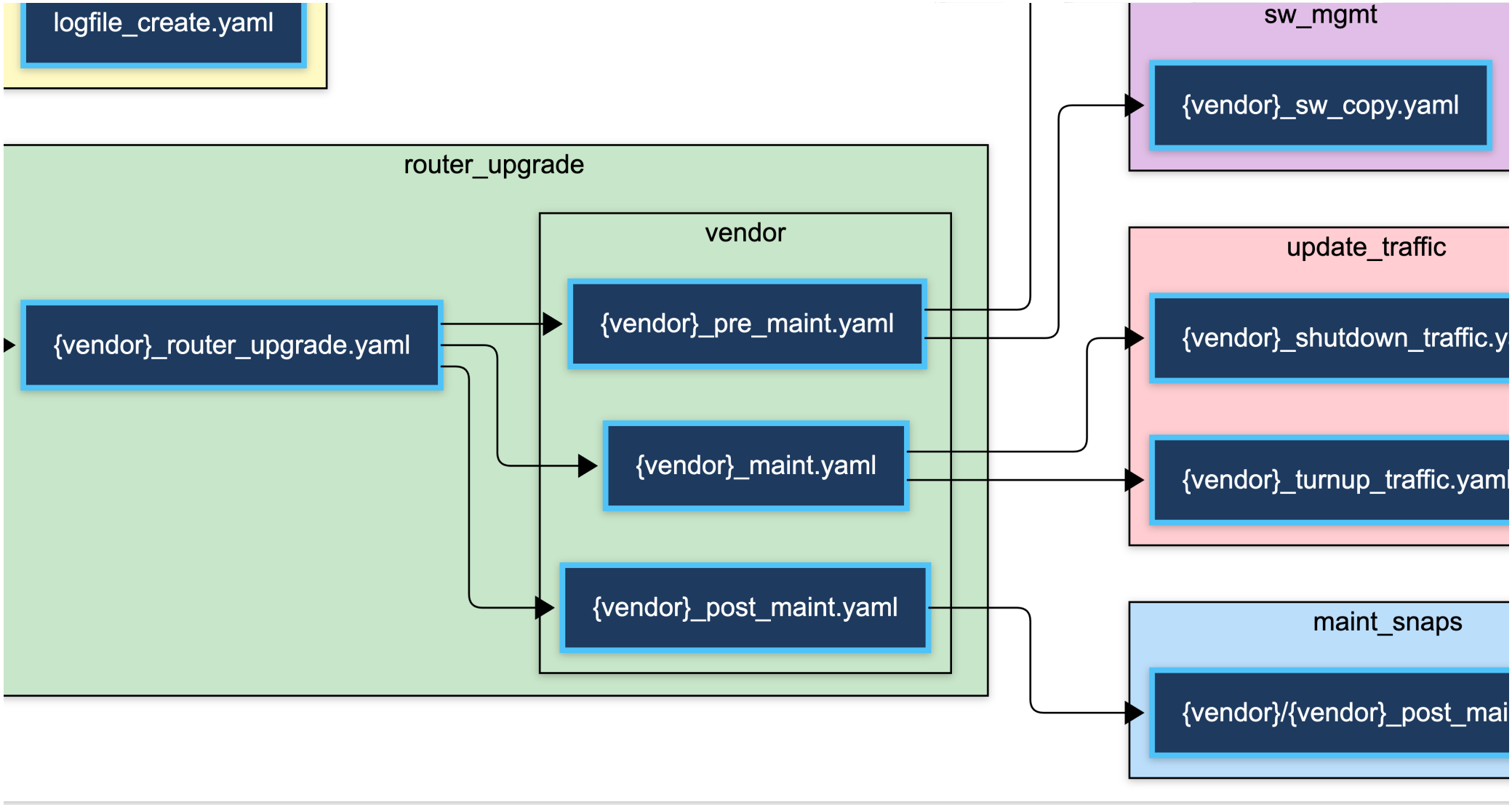


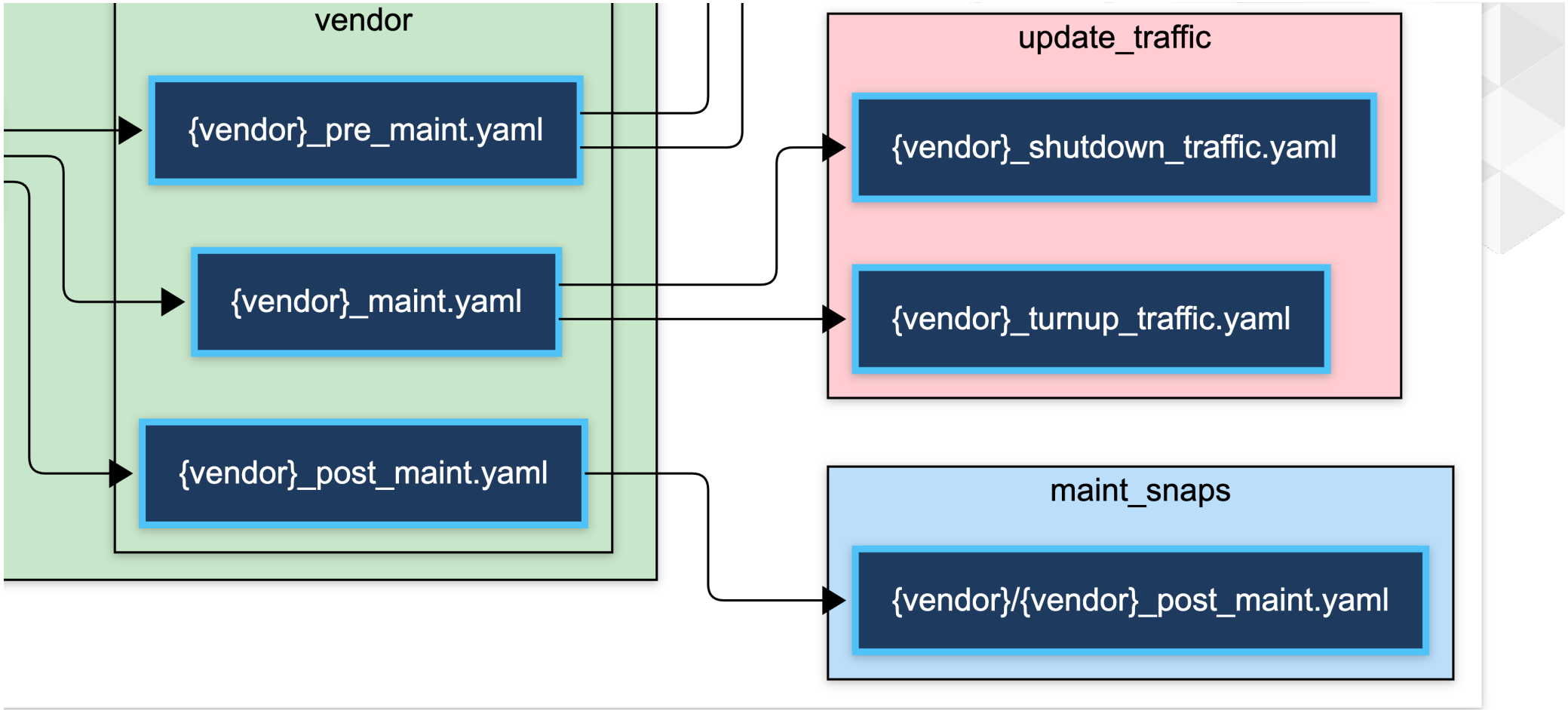


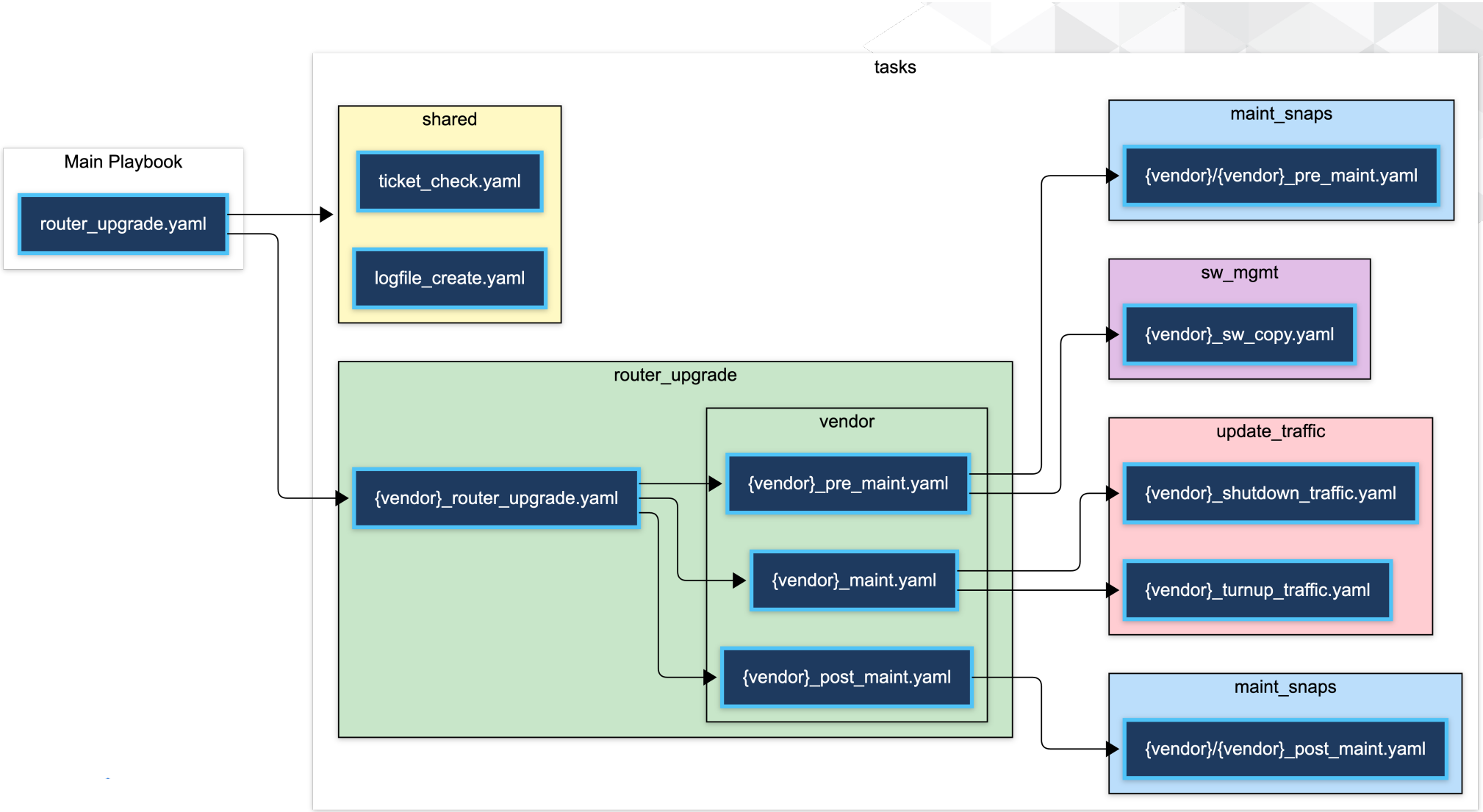

tasks
maint_snaps
{vendor}/{vendor}_pre_maint.yaml
sw_mgmt
{vendor}_sw_copy.yaml
router_upgrade
vendor
{vendor}_pre_maint.yaml
update_traffic
{vendor}_shutdown_traffic.yaml




{vendor}_sw_copy.yaml
_upgrade
vendor
{vendor}_pre_maint.yaml
{vendor}_maint.yaml
{vendor}_post_maint.yaml
update_traffic
{vendor}_shutdown_traffic.yaml
{vendor}_turnup_traffic.yaml
maint_snaps
{vendor}/{vendor}_post_maint.yaml







Include vs Import Tasks

Include – Dynamic reuse

- Tasks files are processed as they are encountered

Import – Static reuse

- Tasks are pre-processed at runtime.

Include vs Import Tasks

--start-at-task *task_name*

- Make task names unique
  - Ex: *vendor* pre-maintenance snaps
- Can't use block or imported task names

Keywords, loops, conditionals

- Variables used by loops must exist at the time the task file is processed. *Include_tasks* is perfect for this.
- Only apply to the tasks inside the *imported_tasks* file

Lessons Learned

Code Update Management

The Good

Allows updates across multiple playbooks at once.

The Bad

Breaking one playbook, breaks many.

CI/CD Pipeline

Forced to learn more advanced usage

Filter changes across branches

Prevent common oopsie!

Managing across branches

Dev vs Prod

Managing across execution sources

CLI vs AWX

Groundhog Day Scenario

Adopt pipeline and AWX API’s earlier

Better repository naming

Say ”Operational Ansible Playbooks” ten times fast!!!

Thank you

Psst...This is where you ask questions.

NANOG™