Event-driven automation and orchestration

@GRNET with Stackstorm

Lefteris Poulakakis
lepou@noc.grnet.gr

what we have

what we have

- ~50 carrier routers

what we have

- ~50 carrier routers
- ~150 access switches

what we have

- ~50 carrier routers
- ~150 access switches
- ~60 datacenter switches

what we have

what we have

- Ansible for config management

what we have

- Ansible for config management
- Git for VCS

what we have

- Ansible for config management
- Git for VCS
- Tools managing network infrastructure and services

and common workflows...

and common workflows…

- deployment of new services

and common workflows...

- deployment of new services
- provisioning of new devices/replacement of faulty ones

and common workflows...

- deployment of new services
- provisioning of new devices/replacement of faulty ones
- software upgrades

...that

...that

- are well defined

...that

- are well defined
- consume time and human effort

…that

- are well defined
- consume time and human effort
- although, can be scripted (eg. in runbooks)

some of our use cases

some of our use cases

- Datacenter switches mass upgrade

some of our use cases

- Datacenter switches mass upgrade
- Zero Touch Provisioning

some of our use cases

- Datacenter switches mass upgrade
- Zero Touch Provisioning
- Auto-deployment of our Ansible repo changes

some of our use cases

- Datacenter switches mass upgrade
- Zero Touch Provisioning
- Auto-deployment of our Ansible repo changes
- Network Ops tasks part of BMS autoprovision

So...

So...

...let's automate

we need a tool that

we need a tool that

- senses changes at the tools or the network

we need a tool that

- senses changes at the tools or the network
- trigger actions based on them

we need a tool that

- senses changes at the tools or the network
- trigger actions based on them
- can abstract actions into complex workflows

we need a tool that

- senses changes at the tools or the network
- trigger actions based on them
- can abstract actions into complex workflows
- interact with the network and our tools

Stackstorm

Stackstorm

- solid base featureset

Stackstorm

- solid base featureset
- Lots of intergration with other tools (ST2 calls them `packs`)

Stackstorm

- solid base featureset
- Lots of intergration with other tools (ST2 calls them packs)
- Support for "standard" wordflow language (Openstack's Mistral)

Stackstorm

- solid base featureset
- Lots of intergration with other tools (ST2 calls them packs)
- Support for "standard" wordflow language (Openstack's Mistral)
- Native intergration with network infrastructure (with NAPALM pack)

Stackstorm

> IF-This-Then-That automation

Sensors

Inbound/Outbound intergration.
Receive/poll for events.

Stackstorm

> IF-This-Then-That automation

Stackstorm

> **IF-This-Then-That automation**

Sensors

Inbound/Outbound intergration.
Receive/poll for events.

Triggers

Result of an activated sensor.

Stackstorm

> **IF-This-Then-That automation**

Sensors

Inbound/Outbound intergration.
Receive/poll for events.

Triggers

Result of an activated sensor.

Actions

Outbound intergrations. A REST API call, an ansible playbook or a custom script.

Rules

Map triggers to actions. Filter against criteria and pass trigger data to the actions run

Stackstorm

> IF-This-Then-That automation

Stackstorm

> **IF-This-Then-That automation**

Rules

Map triggers to actions. Filter against criteria and pass trigger data to the actions run

Workflows

A connected set of actions.

Stackstorm

❝ **IF-This-Then-That automation**

Rules

Map triggers to actions. Filter against criteria and pass trigger data to the actions run

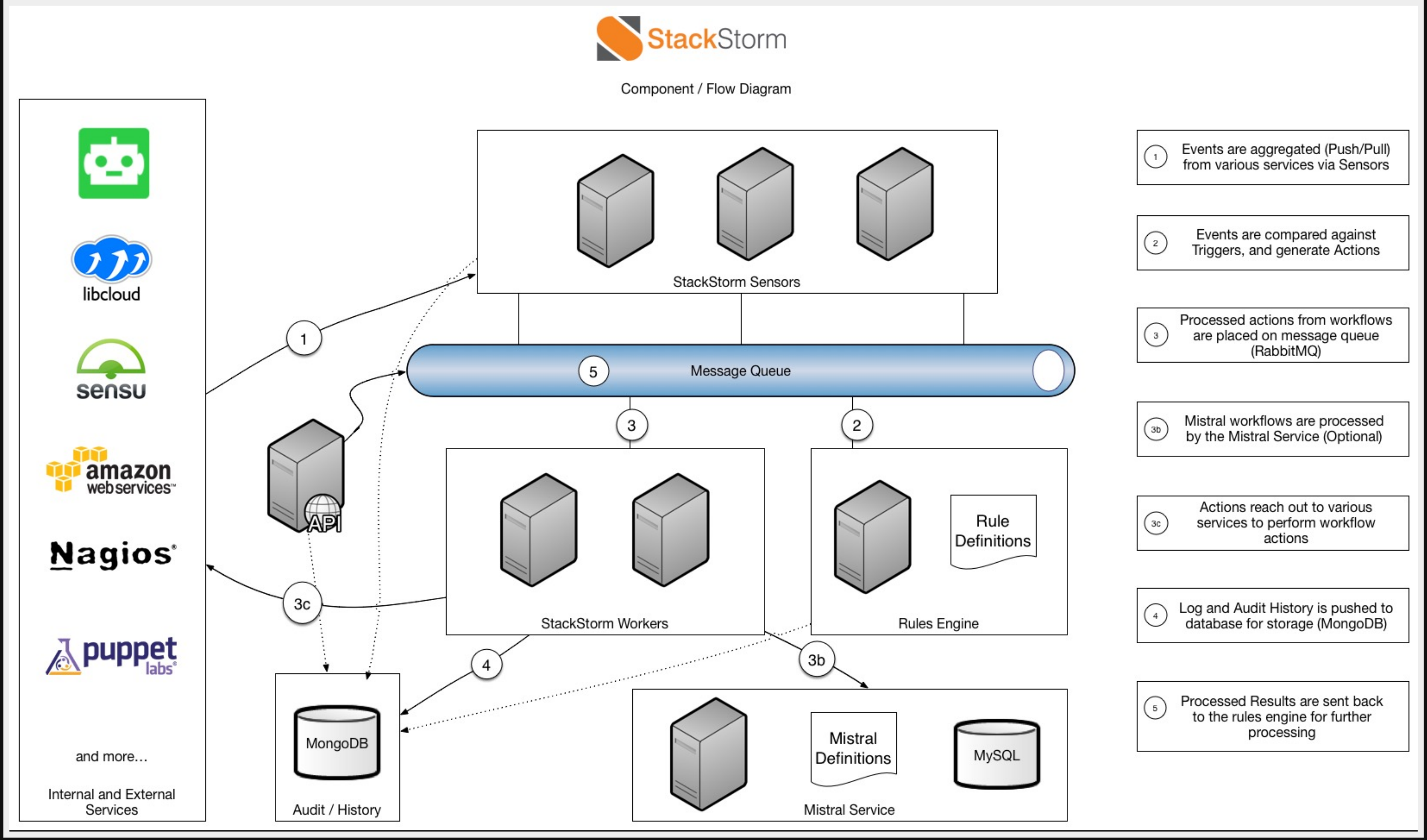
Workflows

A connected set of actions.

Packs

Units of content deployment.
Eq. to a module or a plugin.

Architecture



1 Events are aggregated (Push/Pull) from various services via Sensors

2 Events are compared against Triggers, and generate Actions

3 Processed actions from workflows are placed on message queue (RabbitMQ)

3b Mistral workflows are processed by the Mistral Service (Optional)

3c Actions reach out to various services to perform workflow actions

4 Log and Audit History is pushed to database for storage (MongoDB)

5 Processed Results are sent back to the rules engine for further processing

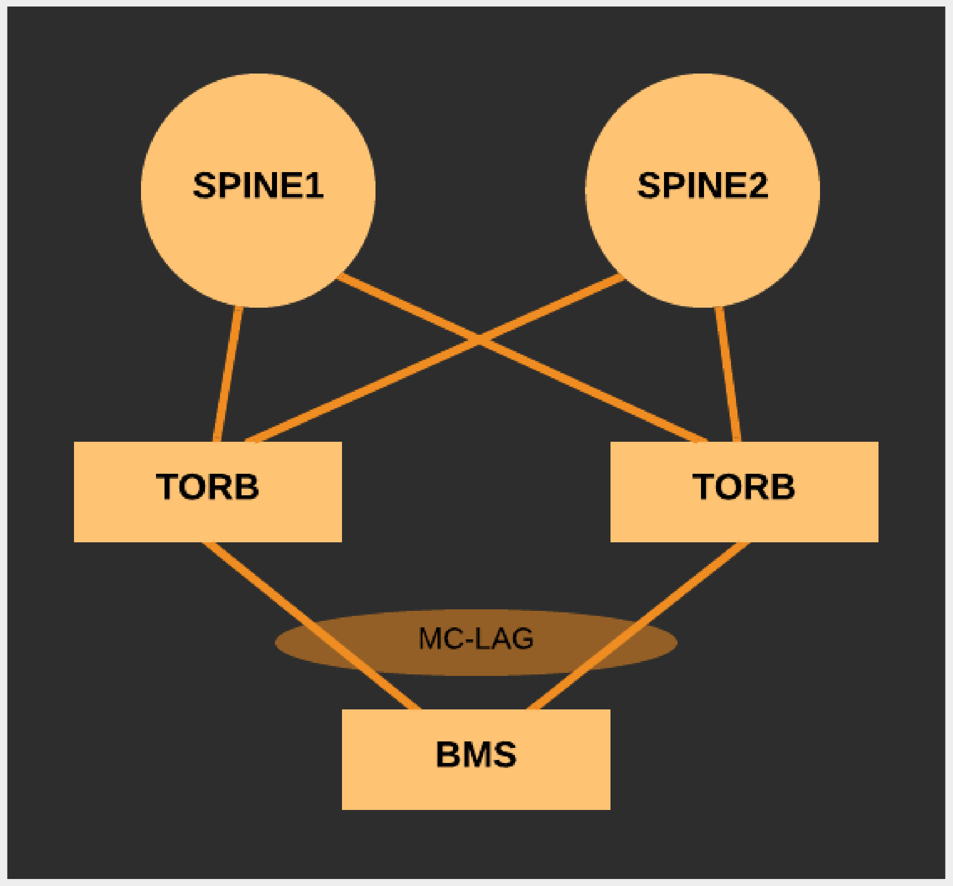
The anatomy of a pack

```
├── actions
│   ├── drain-leaf.meta.yaml
│   ├── evaluate_bgp_peers_status.py
│   ├── evaluate_bgp_peers_status.yaml
│   ├── junos-upgrade.meta.yaml
│   ├── undrain-leaf.meta.yaml
│   ├── upgrade-leaf.meta.yaml
│   └── workflows
│       ├── drain-leaf.yaml
│       ├── junos-upgrade.yaml
│       ├── mistral-leaf-ztp.yaml
│       ├── undrain-leaf.yaml
│       └── upgrade-leaf.yaml
│
```

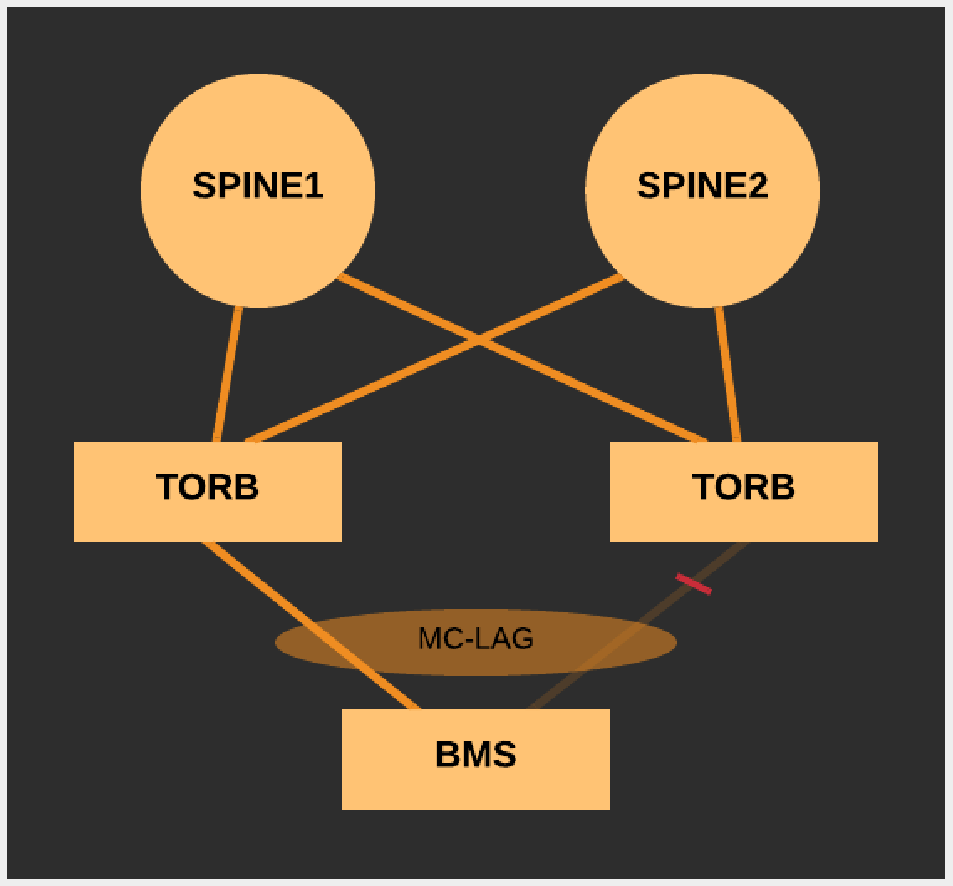
The anatomy of a pack

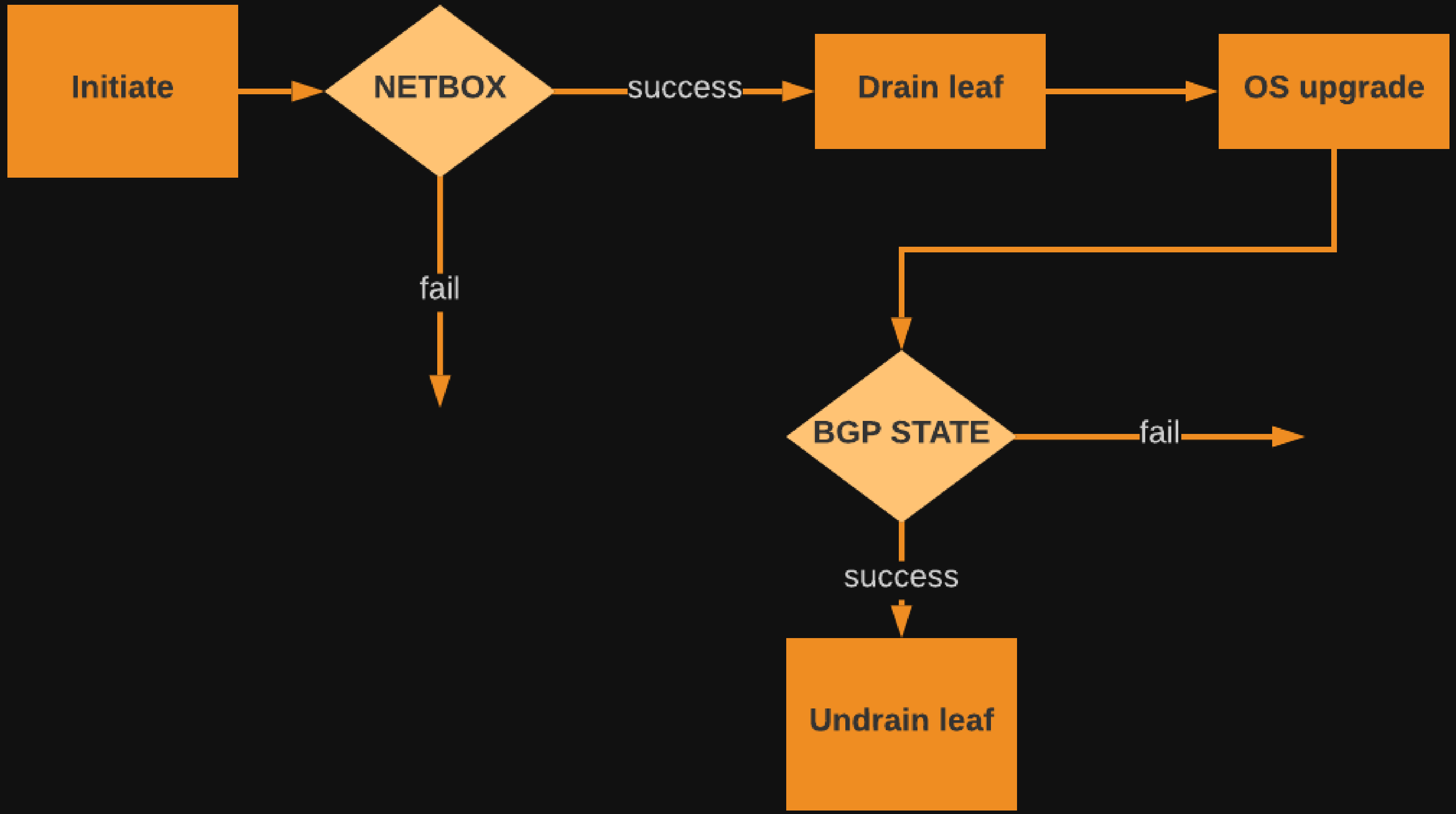
```
|
├── icon.png
├── pack.yaml
├── requirements.txt
└── rules
    ├── drain_leaf.yaml
    ├── fabric_leaf_ztp.yaml
    ├── junos_upgrade.yaml
    ├── undrain_leaf.yaml
    ├── upgrade_fabric.yaml
    └── upgrade_leaf.yaml
```

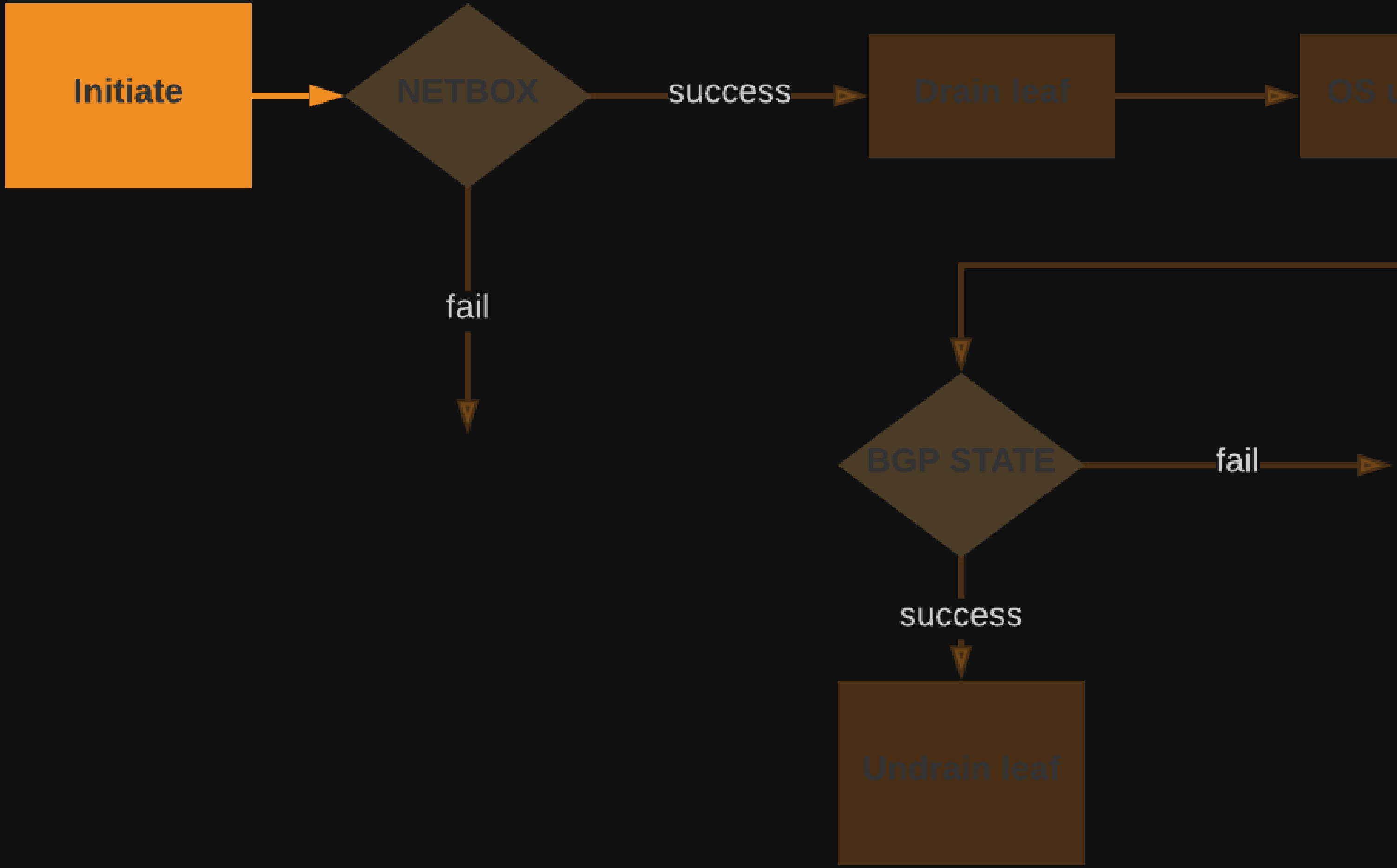
DC switches mass upgrade
Our DCs architecture



DC switches mass upgrade
Our DCs architecture




Initiate
NETBOX
success
fail
Drain leaf
OS upgrade
BGP STATE
fail
success
Undrain leaf



Initiate

NETBOX

success

Drain leaf

OS upgrade

fail

BGP STATE

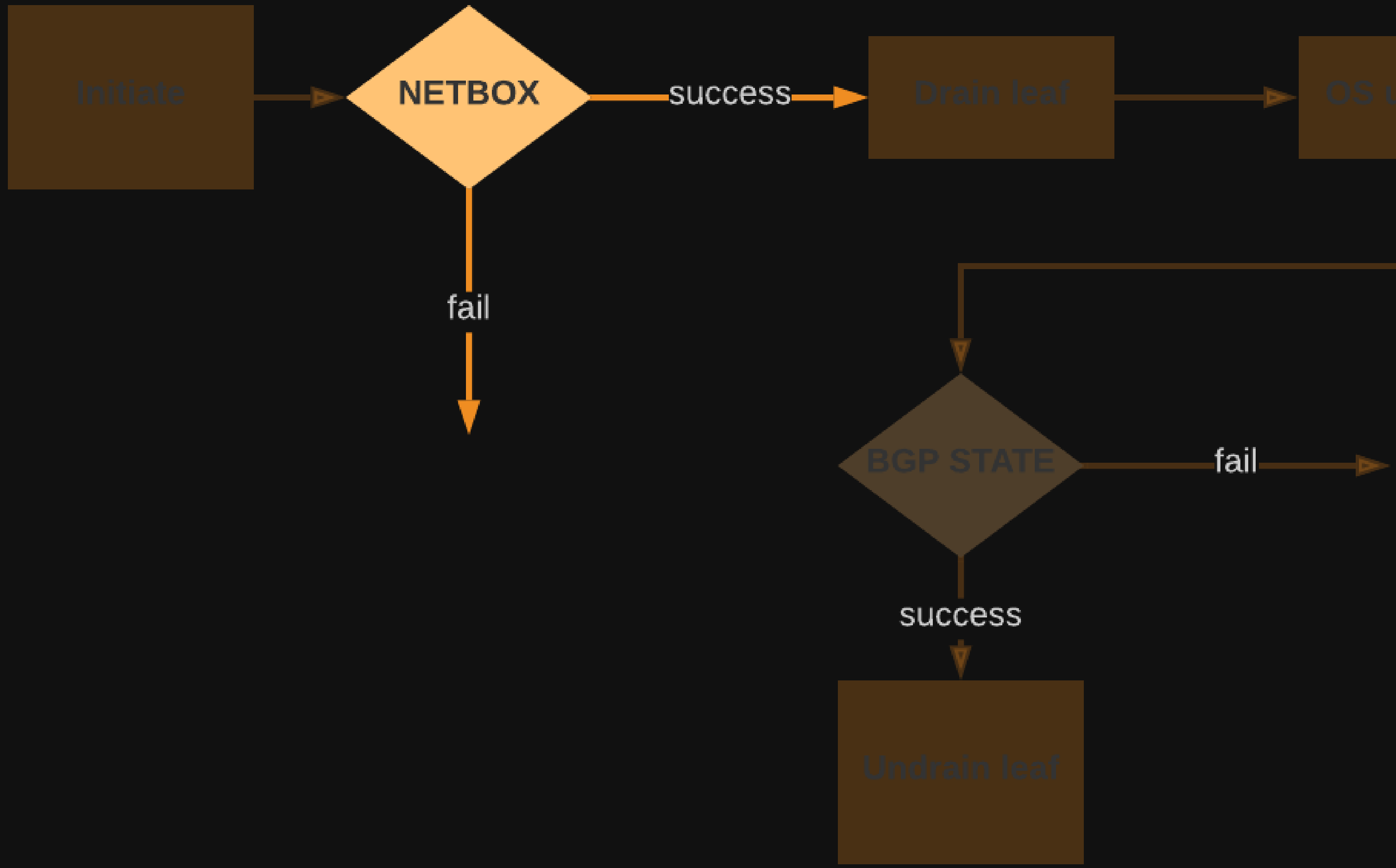
fail

success

Undrain leaf

rules/upgrade-leaf.yaml

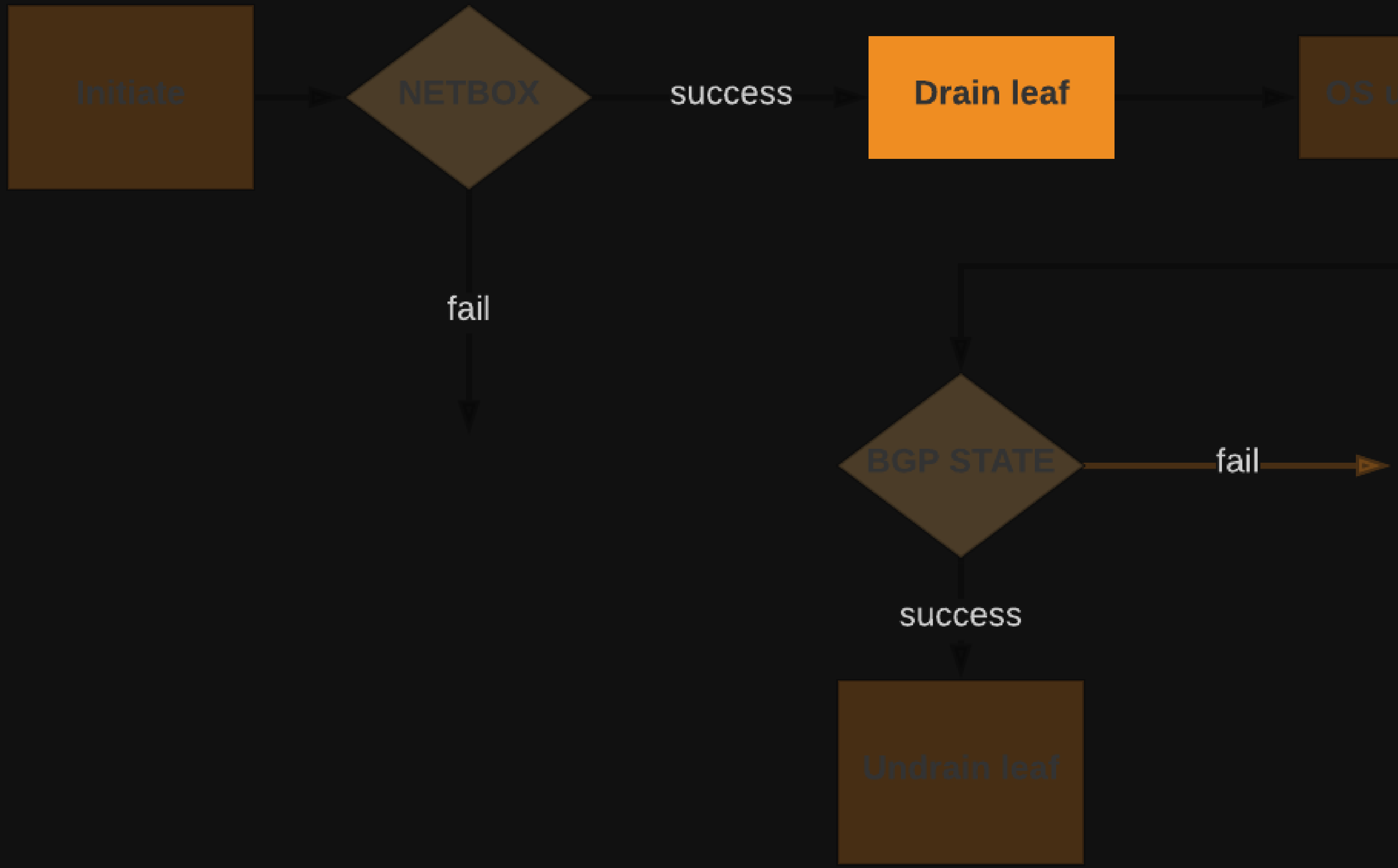
```yaml
name: "upgrade_leaf"
pack: "grnet"
description: "Full workflow of the JunOS upgrade of a QFX5100"
enabled: true
trigger:
  type: "core.st2.webhook"
  parameters:
    url: "upgrade_leaf"
criteria: {}
action:
  ref: "grnet.upgrade-leaf"
  parameters:
    upgrade_data: "{{ trigger.body }}"
```



Initiate
NETBOX
success
fail
Drain leaf
OS upgrade
BGP STATE
fail
success
Undrain leaf

actions/workflows/upgrade-leaf.yaml

```yaml
netbox_device:
   action: netbox.get.dcim.devices
   input:
     name: "{{ _.dev_name }}"
   publish:
     dev_status: "{{ task('netbox_device')..status.label }}"
     dev_role: "{{task('netbox_device')..device_role.slug }}"
   on-success:
     - fail: "{{ _.dev_status != 'Active'
                        or _.dev_role != 'dc-fabric-leaf' }}"
     - bgp_status
     - admin_down_downlinks
```



Initiate

NETBOX

success

Drain leaf

OS upgrade

fail

BGP STATE

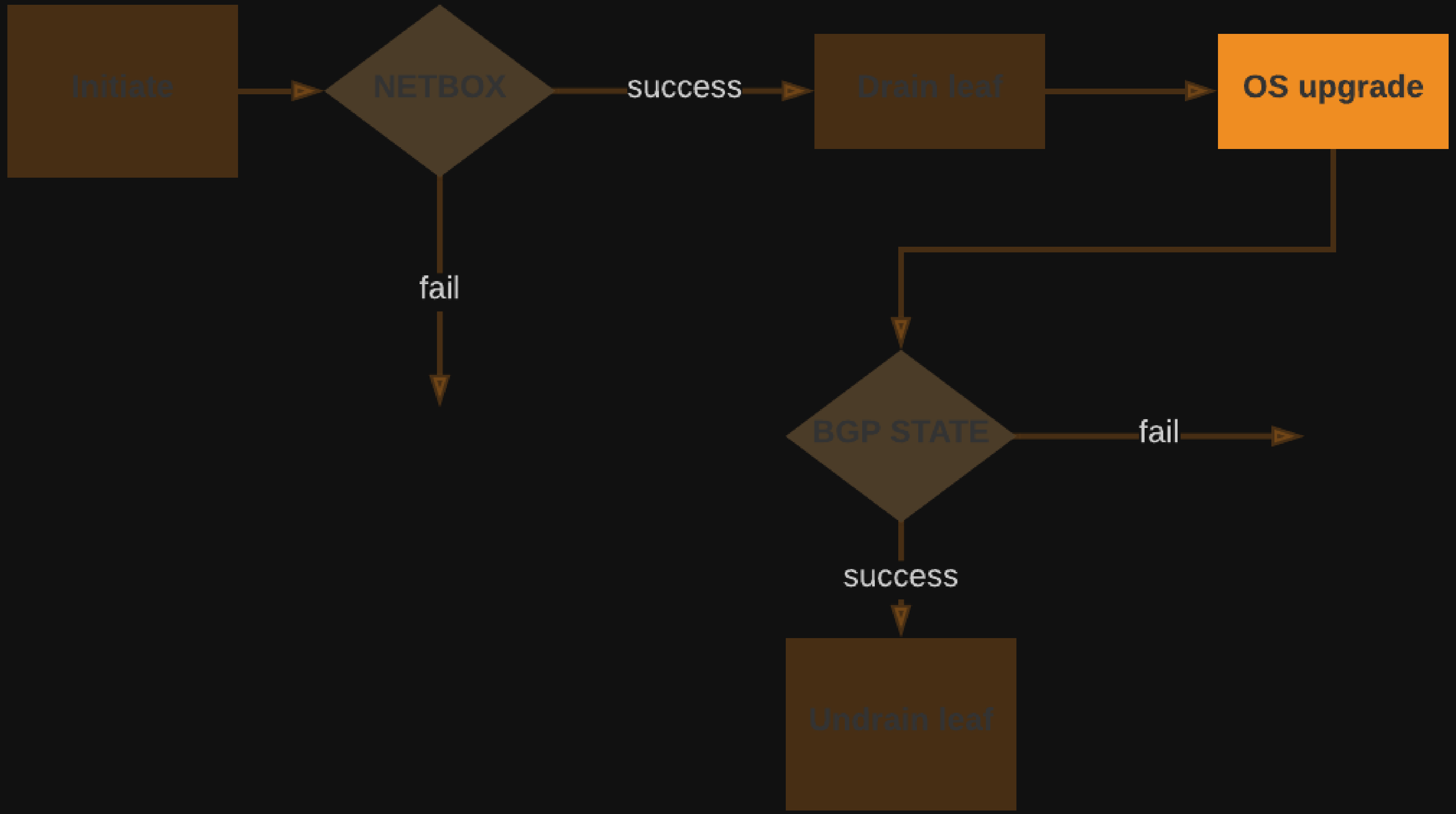
fail

success

Undrain leaf

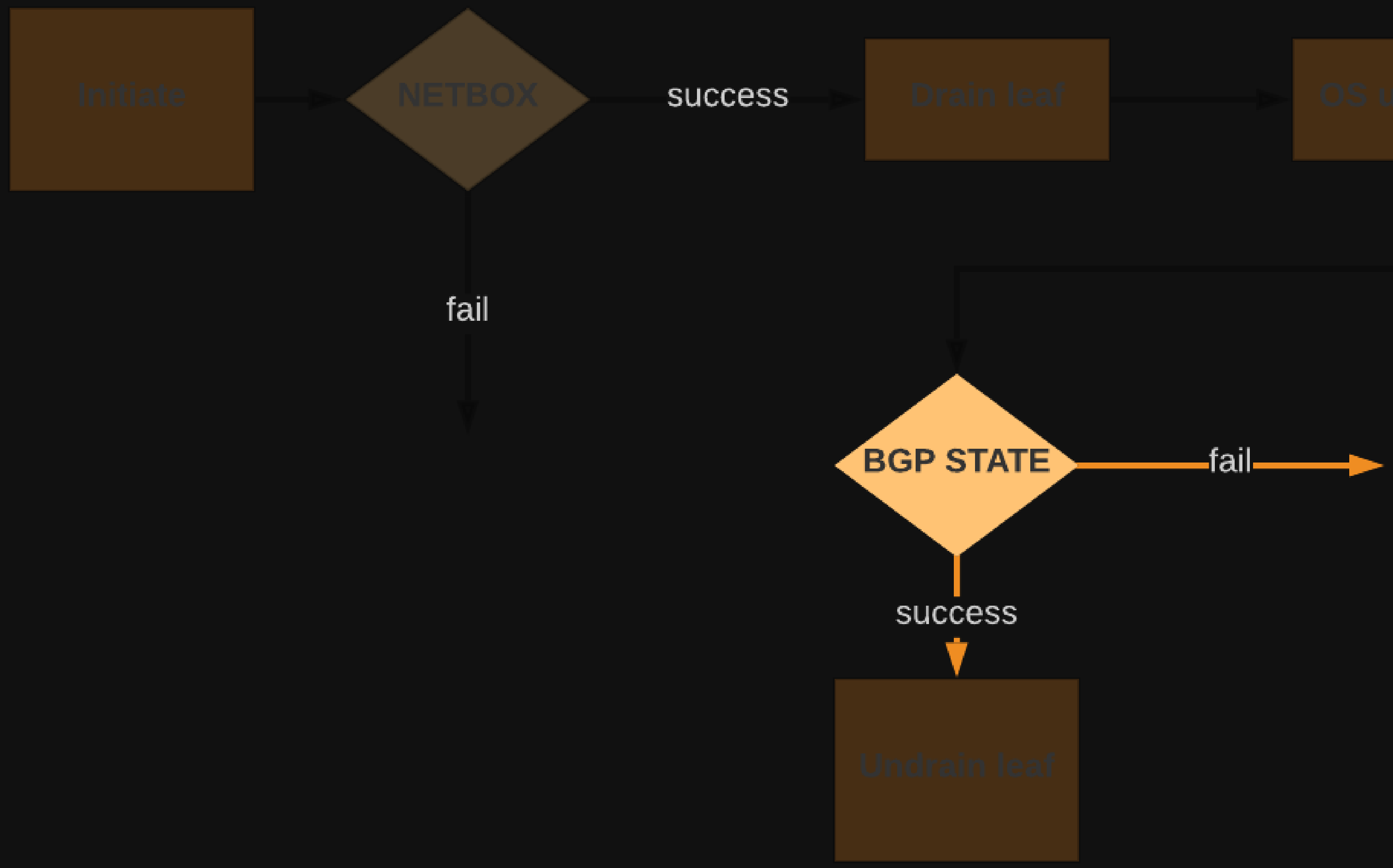
actions/workflows/drain-leaf.yaml

```
admin_down_downlinks:
# Load the configuration that drains the switch
  action: napalm.loadconfig
  input:
    config_file: "/srv/st2/static/drain_leaf_config.set"
    hostname: "{{ _.dev_name }}"
    driver: "junos"
```


Initiate
NETBOX
success
fail
Drain leaf
OS upgrade
BGP STATE
fail
success
Undrain leaf

actions/workflows/junos-upgrade.yaml

```
qfx5100_upgrade:
  action: ansible.playbook
  input:
    playbook: "/srv/ansible/playbooks/junos-install.yaml"
    limit: "{{ _.dev_name }}"
    start_at_task: "Execute a basic Junos software upgrade"
    extra_vars:
      - user: "{{}}"
      - password: "{{ st2kv.system.st2_devices_pass
                        | decrypt_kv }}"
      - force_host: true
      - remote_pkg: "{{ _.junos_url }}"
  on-success:
    - ping_host
```



Initiate

NETBOX

success

fail

Drain leaf

OS upgrade

BGP STATE

fail

success

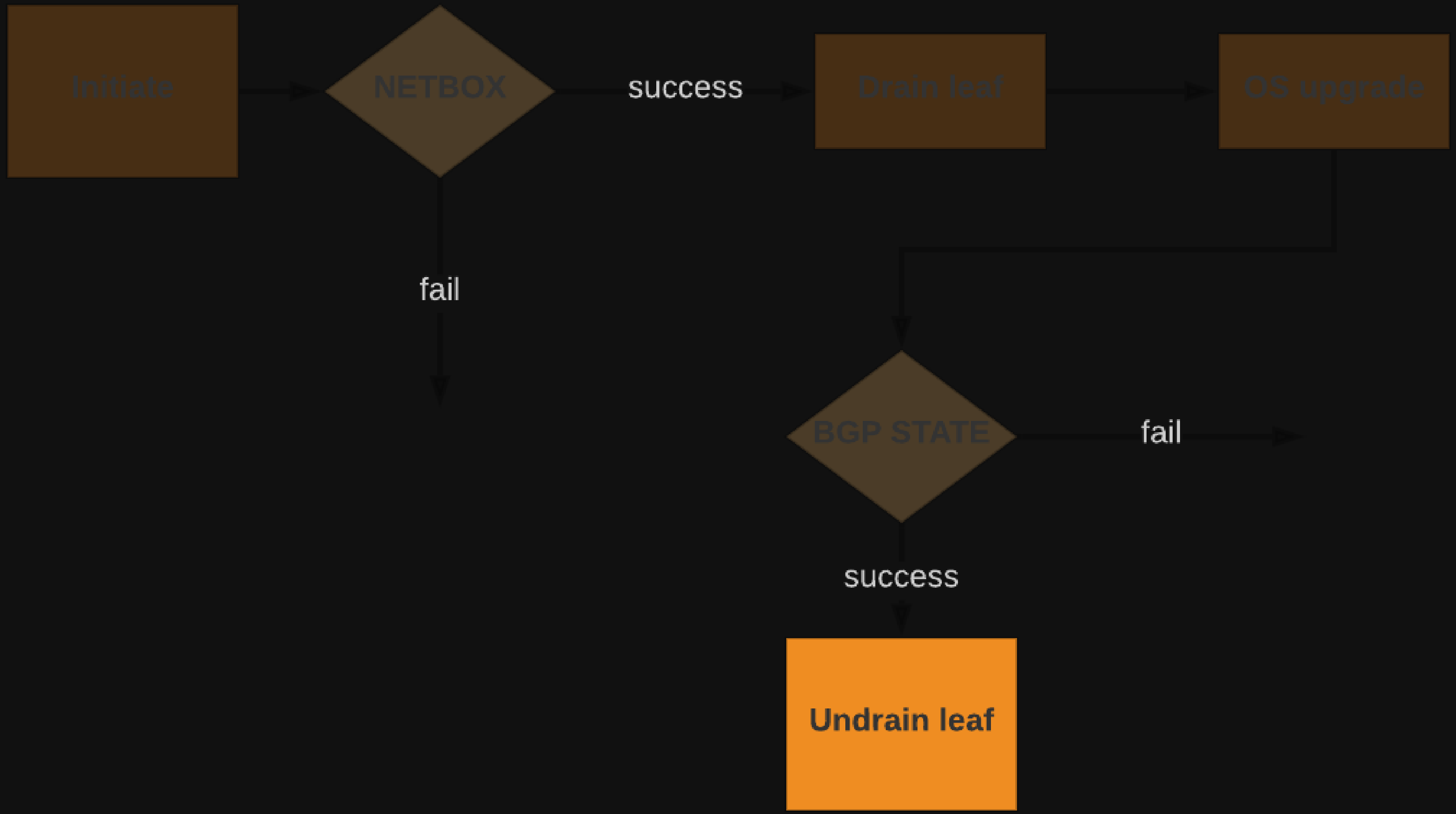
Undrain leaf

actions/workflows/undrain-leaf.yaml

```yaml
bgp_status:
  # Get the BGP status of the device leaf
  action: napalm.get_bgp_neighbors
  input:
    hostname: "{{ _.dev_name }}"
    driver: "junos"
    credentials: "stackstorm"
  on-success:
    - evaluate_bgp_output
```

actions/workflows/undrain-leaf.yaml

```yaml
evaluate_bgp_status:
  action: grnet.evaluate_bgp_peers_status
  input:
    bgp_state: "{{ _.bgp_after_status }}"
  on-success:
    - restore_port_state
```


Initiate
NETBOX
success
Drain leaf
OS upgrade
fail
BGP STATE
fail
success
Undrain leaf

actions/workflows/undrain-leaf.yaml

```yaml
ansible_create_config:
  action: ansible.playbook
  input:
    playbook: "create-config.yml"
  on-success:
    - restore_port_state

restore_port_state:
  join: all
  action: ansible.playbook
  input:
    playbook: "junos-commit-and-confirm.yml"
```

some of our use cases

- Datacenter switches mass upgrade (done for leaf switches)
- Zero Touch Provisioning (done)
- Network Ops tasks part of BMS autoprovision (developing)
- Auto-deployment of our Ansible repo changes (brainstorming)

to sum up

to sum up

- runbooks all the way

to sum up

- runbooks all the way
- automate common tasks

grnet

to sum up

- runbooks all the way
- automate common tasks
- trust automation for more critical tasks

to sum up

- runbooks all the way
- automate common tasks
- trust automation for more critical tasks
- can't automate the human

Questions?