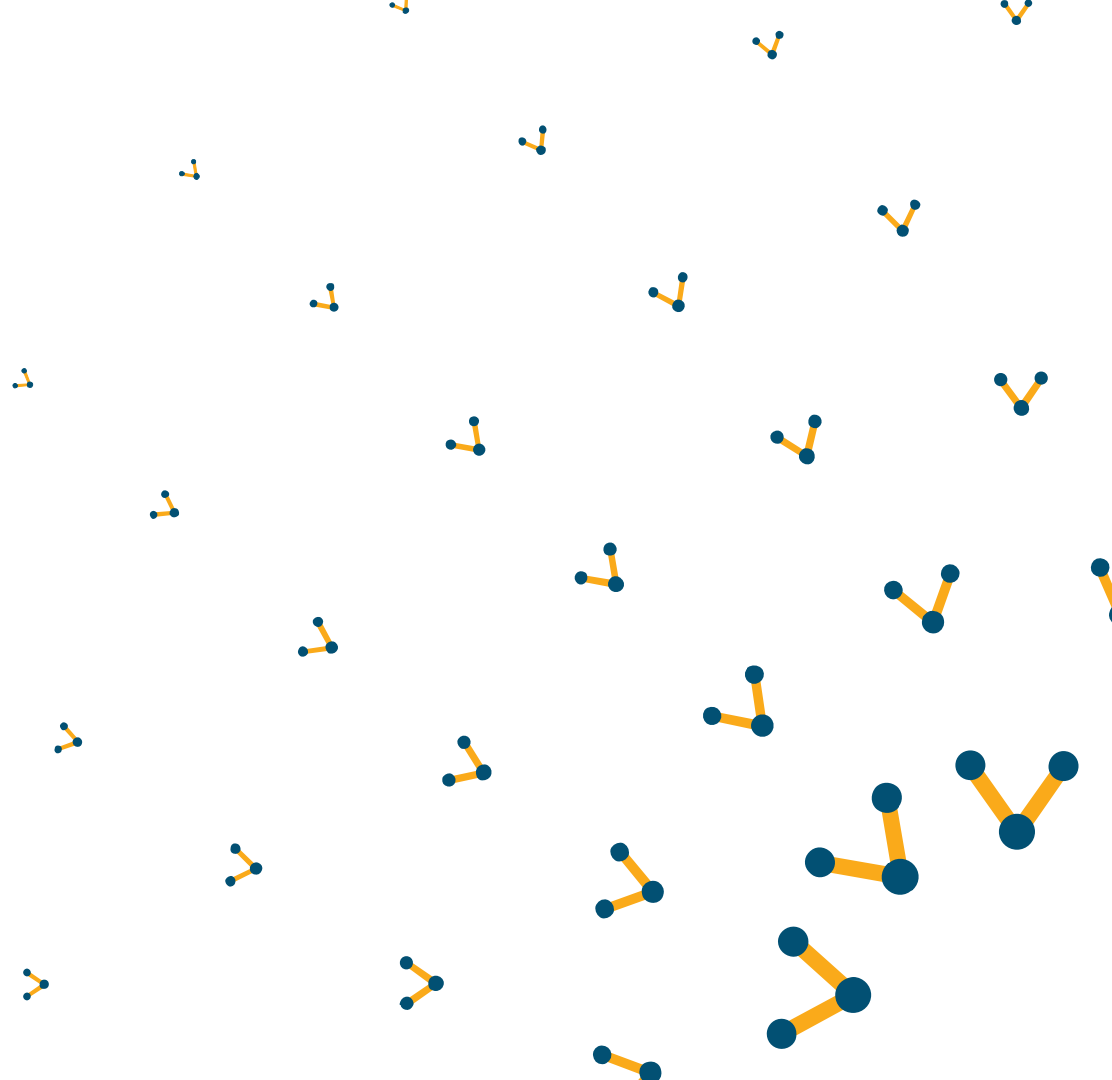
DeveloperDays
Network Services Orchestrator

Feature-flags & Phased rollouts

Controlled service modifications

Kristian Larsson
Architect
2020-04-29

Controlling change



Scenario

- Day 1, you build a service
  - It configures a "backbone" interfaces
- Day 2, deploy service in network
- Day 3, someone requires changes to the service
  - MTU should be 9100 instead of 1500!

... how to change the service?

© 2019 Cisco and/or its affiliates. All rights reserved. Cisco Public

Two approaches

- Naïve approach

- Feature-flag approach

© 2019 Cisco and/or its affiliates. All rights reserved. Cisco Public

Naive approach

1. Change service configuration template
2. `git commit`
3. Deploy new version of your NSO service package
4. `re-deploy` service instance

➔ new config now active in network

© 2019 Cisco and/or its affiliates. All rights reserved. Cisco Public

```
<config-template xmlns="http://tail-f.com/ns/config/1.0">
    <devices xmlns="http://tail-f.com/ns/ncs">

        <device tags="nocreate">
            <name>{/device}</name>
            <config tags="merge">
                <interface-configurations xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-ifmgr-cfg">
                    <interface-configuration>
                        <active>act</active>
                        <interface-name>{/interface}</interface-name>
                        <description>Link to {/remote/device} [{/remote/interface}]</description>
                        <mtus>
                            <mtu>
                                <owner>{$INTERFACE_TYPE}</owner>
                                <!-- new hard-coded MTU -->
                                <mtu>9100</mtu>
                            </mtu>
                        </mtus>
                        <shutdown tags="delete" when="{/shutdown='false'}"/>
                        <!-- ... other config stuff ... -->
                    </interface-configuration>
                </interface-configurations>
            </config>
        </device>
    </devices>
</config-template>
```

© 2019 Cisco and/or its affiliates. All rights reserved. Cisco Public

Testing a naïve change

- Changing value from (default) 1500 to 9100 is **simple**
- No extra test case
  - just check that it works with 9100

- 1 boolean = 2 values -> 2 test cases
- 1^2 = 2
- 2^2 = 4
- 3^2 = 8
- 4^2 = 16
- 5^2 = 32
- 6^2 = 64
- 7^2 = 128
- 8^2 = 256

© 2019 Cisco and/or its affiliates. All rights reserved. Cisco Public


The loaded gun

LOADED GUN

- Gun is loaded from:
  - package deploy
  - ... until...
  - service re-deploy

© 2019 Cisco and/or its affiliates. All rights reserved. Cisco Public

time

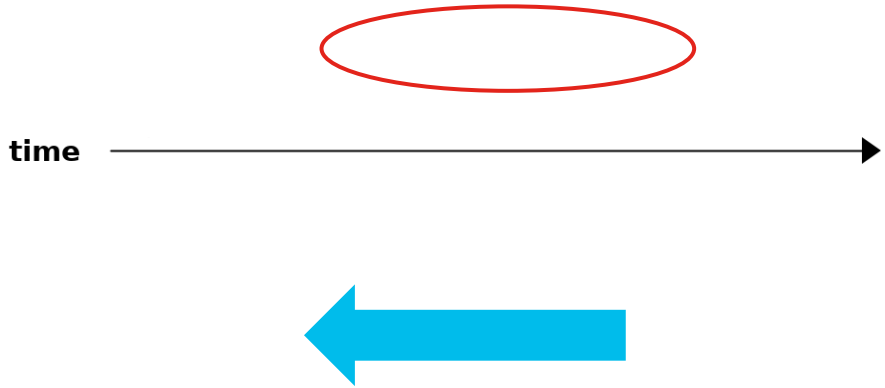
```
andy@nso> configure
Entering configuration mode private
[ok][2020-05-25 11:38:41]

[edit]
andy@nso% set backbone-interface ABC-CORE-1 et-3/2/1
description "Link to FOO-CORE-1"
[ok][2020-05-25 11:38:52]

[edit]
andy@nso% commit
```

© 2019 Cisco and/or its affiliates. All rights reserved. Cisco Public

No revert

- Template change + re-deploy moves forward
- No way back

- Selective rollback is optimistic



© 2019 Cisco and/or its affiliates. All rights reserved. Cisco Public

Feature grouping

- we implement feature A and feature B
- both are merged to master & deploy new NSO package
- service re-deploy deploys both A & B
  - impossible to selectively enable A or B
  - if A or B causes problems we need to roll back both
- BAD: feature A & B have inadvertently been **grouped** together
  - development time is **tightly coupled** to operations

© 2019 Cisco and/or its affiliates. All rights reserved. Cisco Public

Goals

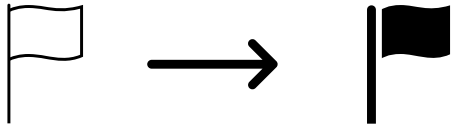
- Allow (reasonable) testing
  - Avoid combinatorial explosion
- No loaded gun
- Going backwards / rollback
- No grouping of features
- Loose coupling between development & operations

© 2019 Cisco and/or its affiliates. All rights reserved. Cisco Public

Feature flags

Feature-flags

- well known concept in software development
- move introduction of change from commit/deploy time to run time
  - temporal decoupling of development and operations!!!
- focus on transition / change
  - limited life time



© 2019 Cisco and/or its affiliates. All rights reserved. Cisco Public

```
list backbone-interface {
  key "device interface";
  // other things

  container feature-flags {
    leaf high-mtu {
      type boolean;
      description "Enable new high MTU (9100). Disable for old MTU (1500)";
      default "false";
    }
  }
}
```

Emphasize old -> new transition

© 2019 Cisco and/or its affiliates. All rights reserved. Cisco Public

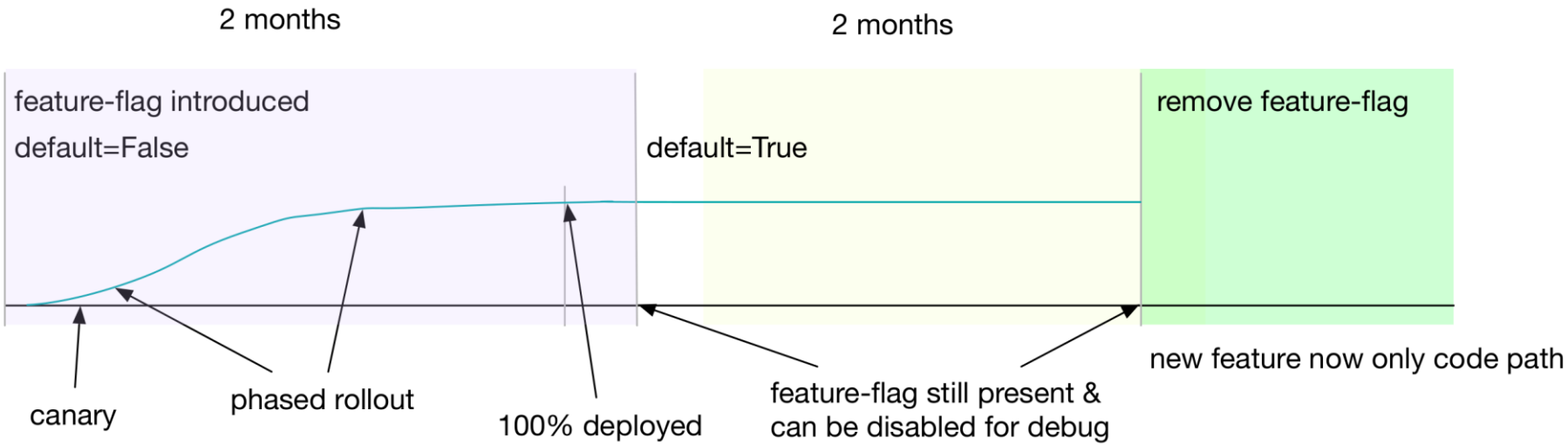
```
<config-template xmlns="http://tail-f.com/ns/config/1.0">
    <devices xmlns="http://tail-f.com/ns/ncs">

        <device tags="nocreate">
            <name>{/device}</name>
            <config tags="merge">
                <interface-configurations xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-ifmgr-c
                    <interface-configuration>
                        <active>act</active>
                        <interface-name>{/interface}</interface-name>
                        <description>Link to {/remote/device} [{/remote/interface}]</descripti
                        <mtus>
                            <mtu>
                                <owner>{$INTERFACE_TYPE}</owner>
                                <!-- new high MTU conditioned on feature-flags -->
                                <mtu when="/feature-flags/high-mtu='true'">9100</mtu>
                            </mtu>
                        </mtus>
                        <shutdown tags="delete" when="{/shutdown='false'}"/>
                        <!-- ... other config stuff ... -->
                    </interface-configuration>
                </interface-configurations>
            </config>
        </device>
    </devices>
</config-template>
```

Sociotechnical

- technically, FF is *just another input*
- NSO won't treat it differently
- difference is in concept
  - clear life cycle for FF
  - introduce FF for change transition
  - when done, **remove** FF
    - keeps down input / permutations over time

© 2019 Cisco and/or its affiliates. All rights reserved. Cisco Public

Feature-flag life cycle



© 2019 Cisco and/or its affiliates. All rights reserved. Cisco Public

Anti-pattern

- could add MTU leaf
- allows any value in range
- BAD – many test cases
- Reduce choices!
  - 1500 or 9100!

```
list backbone-interface {
  key "device interface";
  // other things

  leaf mtu {
    type uint16 {
      range "1500..9100";
    }
    description "MTU of service";
    default "1500";
  }
}
```

© 2019 Cisco and/or its affiliates. All rights reserved. Cisco Public

Anti-pattern

- Reduction to choice of 1500 or 9100
- Still, new config knob
- Over time, new knobs lead to combinatorial explosion
- Focus on transition!

```
list backbone-interface {
  key "device interface";
  // other things

  leaf mtu {
    type uint16 {
      range "1500 | 9100";
    }
    description "MTU of service, either
           1500 (old) or 9100 (new)";
    default "1500";
  }
}
```

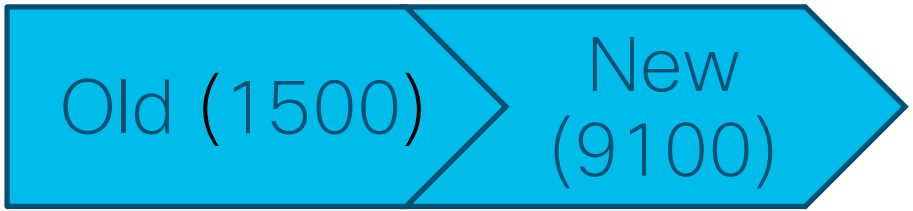
© 2019 Cisco and/or its affiliates. All rights reserved. Cisco Public

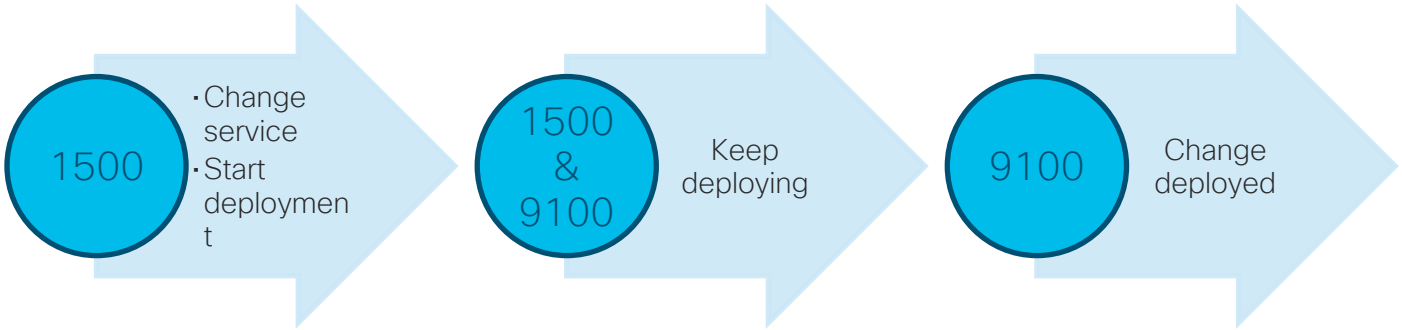
Feature-flag!

- Boolean choice
- Focus on transition

```
1  list backbone-interface {
2    key "device interface";
3    // other things
4
5    container feature-flags {
6      leaf high-mtu {
7        type boolean;
8        description "Enable new high M
9        default "false";
10     }
11   }
12 }
```

© 2019 Cisco and/or its affiliates. All rights reserved. Cisco Public





© 2019 Cisco and/or its affiliates. All rights reserved. Cisco Public

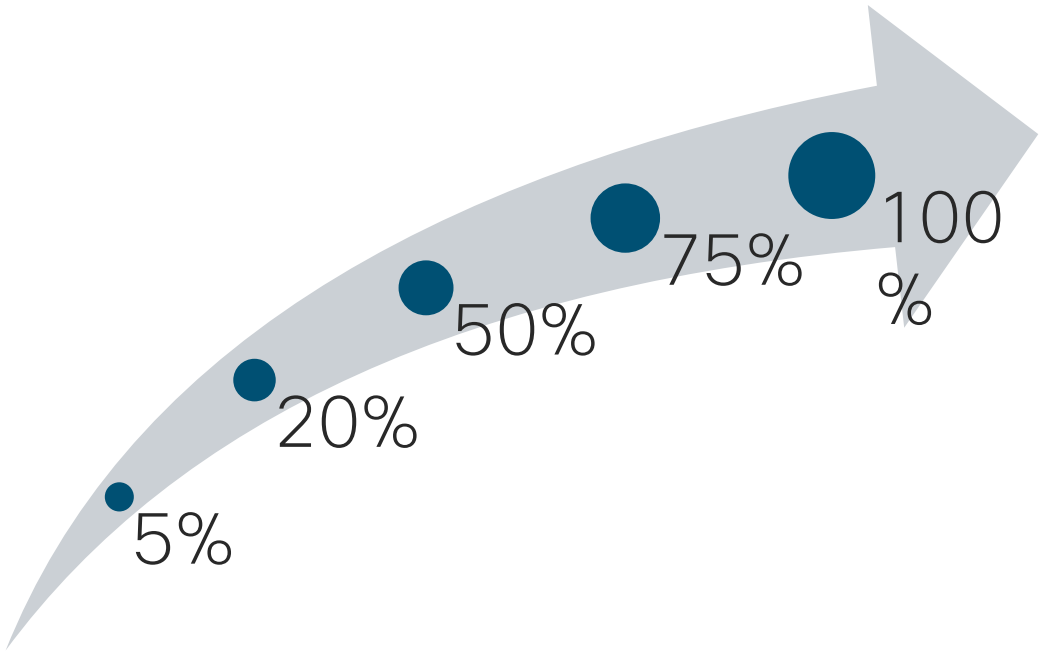
Summary

- Change is natural in life cycle of a service
- For robust operations we must have control of change
- Use feature-flags to gain control
- Focus on:
  - Simple choice
  - Transitional nature

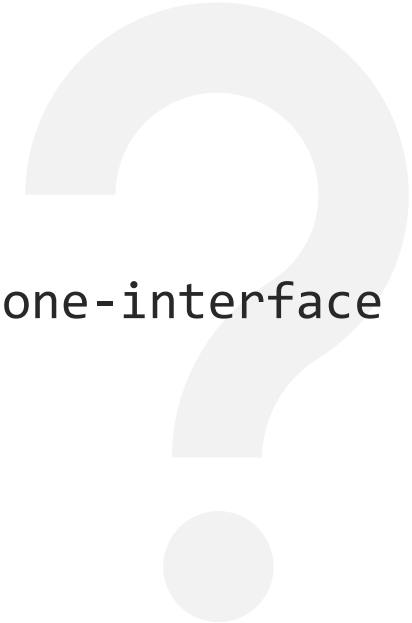
© 2019 Cisco and/or its affiliates. All rights reserved. Cisco Public

Phased rollouts

Phased rollout of interface service



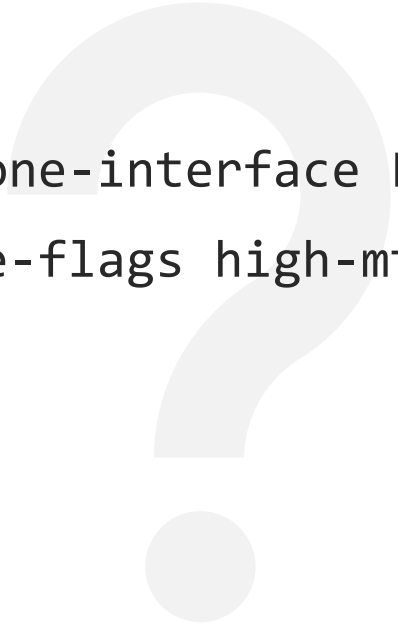
© 2019 Cisco and/or its affiliates. All rights reserved. Cisco Public



```
request backbone-interface * * re-deploy
```

© 2019 Cisco and/or its affiliates. All rights reserved. Cisco Public

```
kll@ncs> configure
kll@ncs% edit backbone-interface FOO-CORE-1 et-3/2/1
kll@ncs% set feature-flags high-mtu true
kll@ncs% commit
[ok]
kll@ncs%
```



© 2019 Cisco and/or its affiliates. All rights reserved. Cisco Public

Errors & configuration validity

- configuration commit only includes syntax and semantic checks
  - an empty configuration is valid
    - but would lead to unhealthy device / service

© 2019 Cisco and/or its affiliates. All rights reserved. Cisco Public

© 2019 Cisco and/or its affiliates. All rights reserved. Cisco Public

Service health

- need to understand if service is **healthy**
- monitor operational state of service
  - is BGP neighbor up?
  - is interface up?
  - can we ping?
- service specific! not generic…

© 2019 Cisco and/or its affiliates. All rights reserved. Cisco Public

Service health via self-test

- YANG action
- Commonly called "self-test"
- Placed under service
- Return common data structure
  - Can return service specific things too
  - Must return "success" leaf
    - In my example...

```
list backbone-interface {
  key "device interface";
  // other stuff

  action self-test {
    tailf:info "Perform self-test of the service";
    tailf:actionpoint "backbone-interface-self-test";
    output {
      leaf success {
        type boolean;
      }

      container interface {
        // service specific health / state about the interface
      }
      container is-is {
        // service specific health / state about IS-IS
      }
      container pim {
        // service specific health / state about PIM
      }
    }
  }
}
```

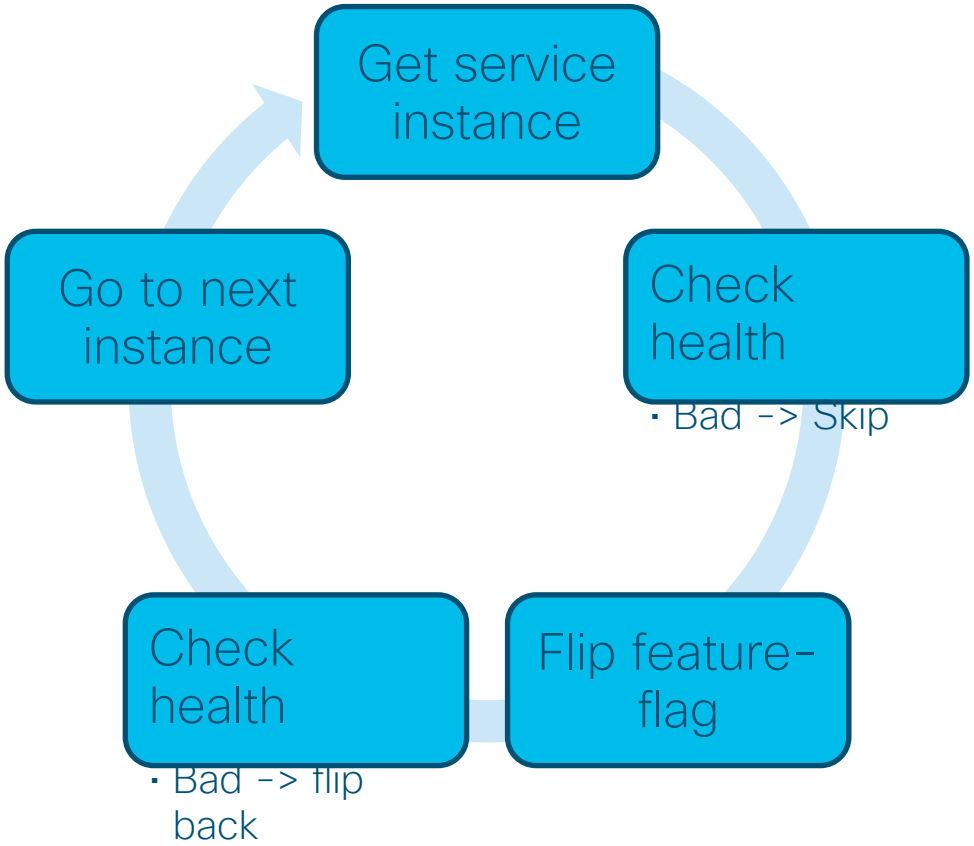
© 2019 Cisco and/or its affiliates. All rights reserved. Cisco Public

```python
class Selftest(Action):
    @Action.action
    def cb_action(self, uinfo, name, kp, action_input, action_output, trans):
        service = ncs.maagic.get_node(trans, kp)
        log.info("self-test for {} {}".format(service.device, service.interface))
        dev = root.devices.device[service.device]
        os = utils.get_dev_os(dev)

        action_output.success = False

        if os == DeviceOs.SROS_CLI:
            intf = dev.live_status.router['Base'].interface[service.interface]
            if intf.oper_state == "Up":
                action_output.success = True
```

- … and some Python to back it up
- Read live-status from device
- Evaluate operational state
- Set success leaf and return

© 2019 Cisco and/or its affiliates. All rights reserved. Cisco Public

Procedure



x400

© 2019 Cisco and/or its affiliates. All rights reserved. Cisco Public

Feature-flag flipper

- Flipping feature-flags is a monotonous task
- … a task for a computer

- I think there should be a package to help out
  - Summarizing feature-flag rollout
  - Flipping feature-flags in controlled manner

© 2019 Cisco and/or its affiliates. All rights reserved. Cisco Public

```
show feature-flags feature-flags
| feature-flag                                                | type           | progress |
|-------------------------------------------------------------+----------------+----------|
| /infrastructure/base-config/feature-flags/foobar            | false-to-true |      73% |
| /infrastructure/backbone-interface/feature-flags/bar | false-to-true |      14% |

show feature-flags instances
| instance                                                                  | type           | value |
|---------------------------------------------------------------------------+----------------+-------+
| /infrastructure/base-config{901-R1-2053}/feature-flags/foobar             | false-to-true | false |
| /infrastructure/base-config{901-R1-2054}/feature-flags/foobar             | false-to-true | true  |
| /infrastructure/bb-intf{901-R1-2053 et-9/0/0}/feature-flags/bar   | true-to-false | false |
| /infrastructure/bb-intf{901-R1-2053 et-10/0/0}/feature-flags/bar | true-to-false | true  |
```

A mock-up of a feature-flag navigator

© 2019 Cisco and/or its affiliates. All rights reserved. Cisco Public

Summary

- Change is natural in life cycle of a service
- For robust operations we must have control of change
- Use feature-flags to gain control
  - Simple choice
  - Transitional nature

- Phased rollouts through service health
- Automate rollout & flipping feature-flags

© 2019 Cisco and/or its affiliates. All rights reserved. Cisco Public


CISCO

Backup slides / details

FF Placement

FF placement

- Where to place feature-flags?

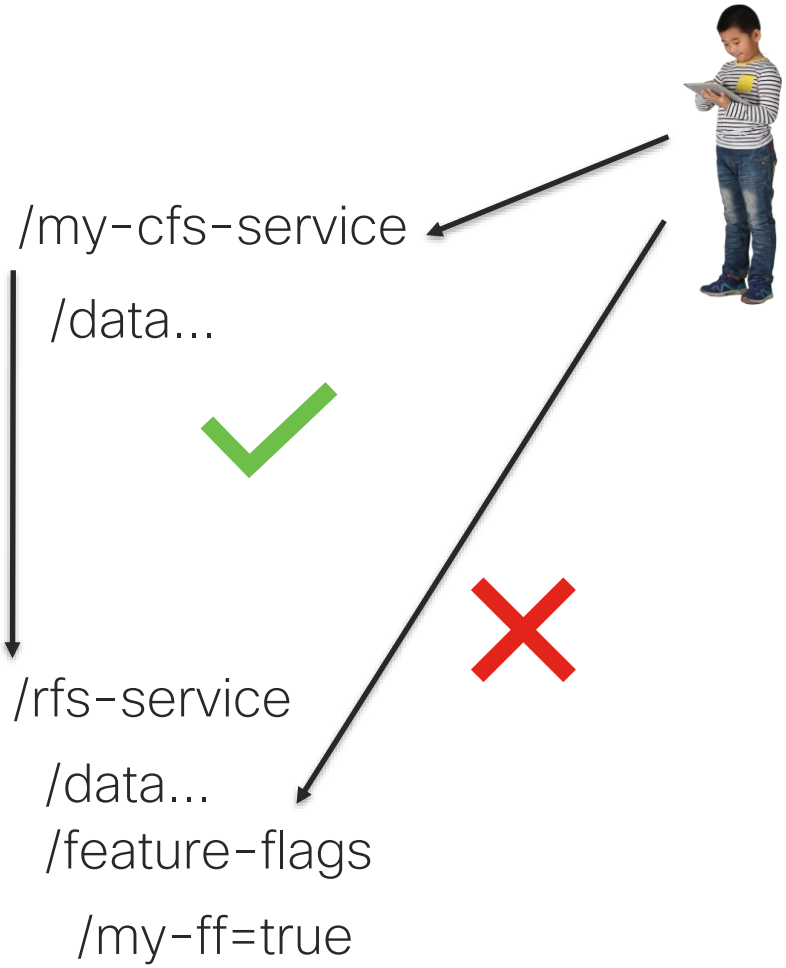
- Under service
  - What I've shown so far

- Separate config tree

© 2019 Cisco and/or its affiliates. All rights reserved. Cisco Public

FF in stack

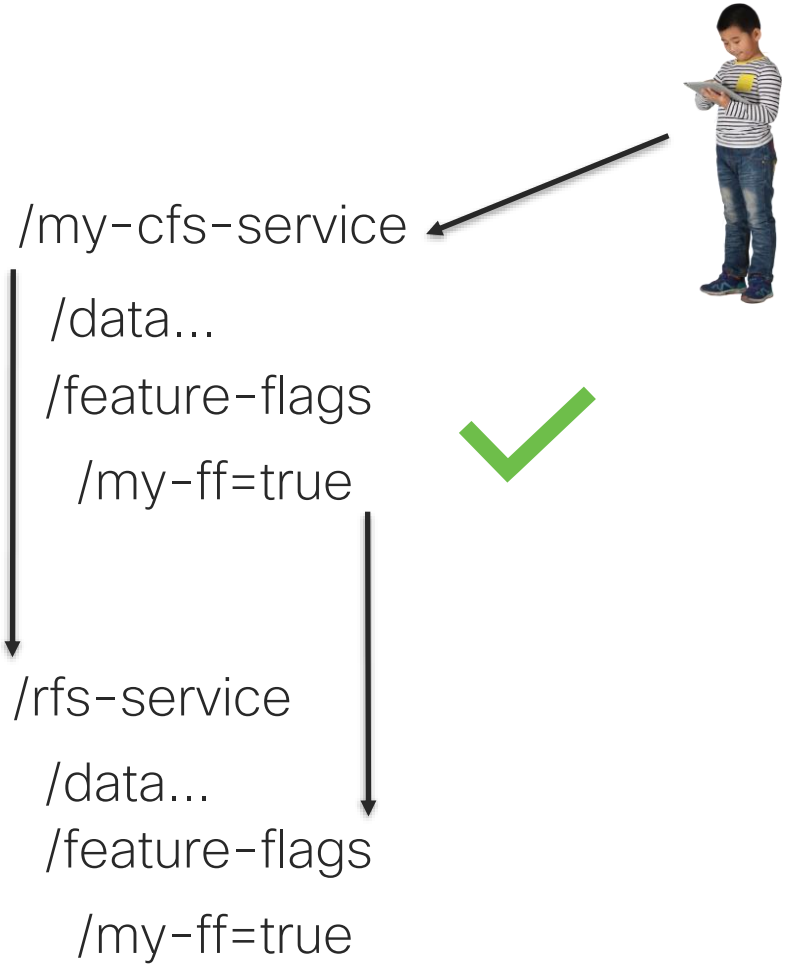
- Avoid user directly modifying RFS that was created by CFS
- Expose FF in CFS!



© 2019 Cisco and/or its affiliates. All rights reserved. Cisco Public

FF in stack

- Avoid user directly modifying RFS that was created by CFS
- Expose FF in CFS!



© 2019 Cisco and/or its affiliates. All rights reserved. Cisco Public

Separate FF tree

- Place FF in separate tree
  - Avoids refcount/ownership issues
- Use kickers to trigger service re-deploy

© 2019 Cisco and/or its affiliates. All rights reserved. Cisco Public