Developer Days
Automation

The bridge to possible

Crosswork Router Automation Kit: Golden Config Templates

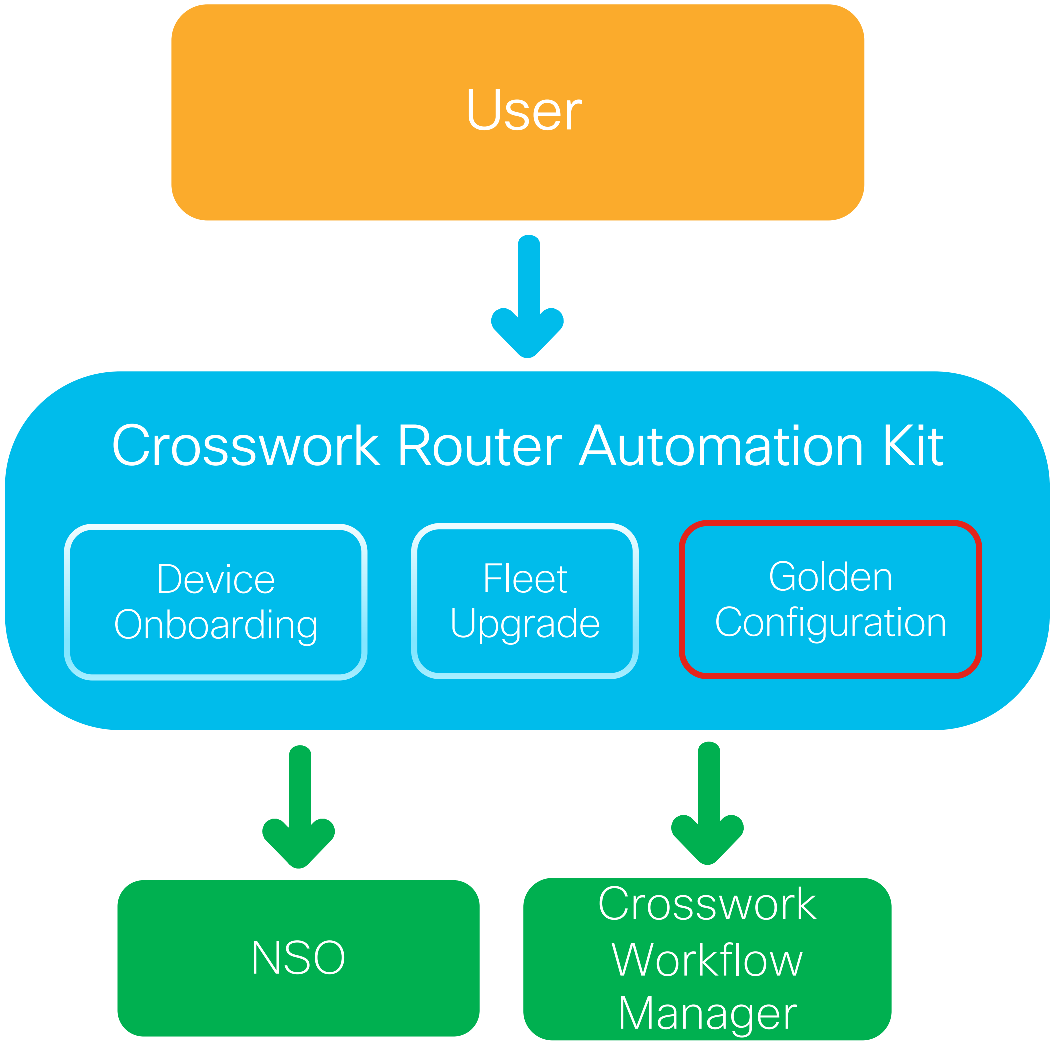
December 6, 2023
One Penn Plaza, NY

Wai Tai, Senior Software Architect
Ravichandran Venkataraman, Technical Marketing Engineering Technical Leader

Agenda

- RAK: Golden Config Templates
- Templates in networking
- Templates in NSO
  - Quick Recap: Device Templates
- Golden Config Templates
  - Why
  - What
  - How
  - Together with Crosswork Workflow Manager
- Questions

Crosswork Router Automation Kit



What is a template in networking domain?

- A predefined configuration or blueprint used for setting up and managing network devices and services.
- Templates are designed to simplify and standardize the deployment and management of network configurations, ensuring consistency and efficiency in network operations.

Templates in NSO

Service Templates

- Part of NSO packages
- Gets loaded upon NSO startup

Device Templates

- Dynamically created by operator as needed and stored as NSO configuration
- Applied on a device using action *apply-template*

Quick Recap:

Device Templates

Device Templates

- Created as part of NSO configuration for manipulating config data in the device tree

```
ncs(config)# show full-configuration devices template
devices template snmp1
 ned-id cisco-ios-cli-3.8
  config
   ios:snmp-server community {$COMMUNITY} RO
   !
  !
 !
```

```
ncs(config)# devices device ce2 apply-template template-name \
      snmp1 variable { name COMMUNITY value 'FUZBAR' }
ncs(config)# show configuration
devices device ce2
 config
  ios:snmp-server community FUZBAR RO
 !
!
ncs(config)# commit dry-run outformat native
native {
    device {
        name ce2
        data snmp-server community FUZBAR RO
    }
}
ncs(config)# commit
Commit complete.
```

Why Golden Config Templates?

- Take advantage of capabilities in Device & Service Templates
- Add support of Jinja2 template engine
- Allow the same configuration to be easily applied through automation across multiple devices on the network resulting in consistent, compliant and accurate device configurations
- Provide support for check-sync, dry-run, FastMap, audit and remediation
- Provide support through Workflows to manage and apply device templates, schedule configuration audit, and remediation

What is it?

Template

```
golden-config {
  template xr-bgp-native {
    tag [ bgp cisco-iosxr day0];
    version 1 {
      type native;
      config










     variable VRF_NAME {
           value testXR;
       }
     }
   }
}
```

Service

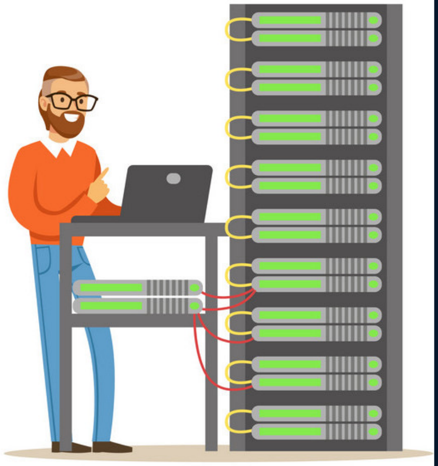
```
golden-config {
  application app-xr-bgp {
    device xr0;
    jinja-template {
       template xr-bgp-native;
       version  1;
    }
    variable AS { value 200; }
    variable AS_REMOTE {
       value 20;
    }
    variable NE { value 20; }
  }
}
```

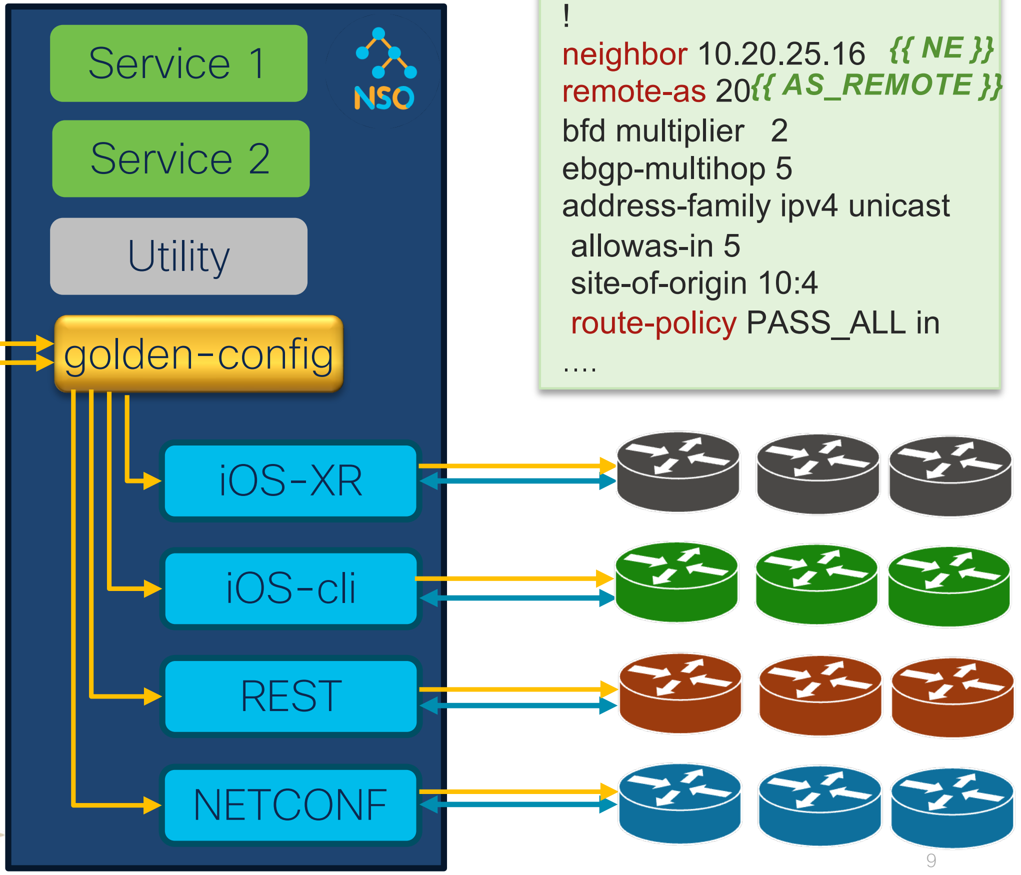
How to use?

Config

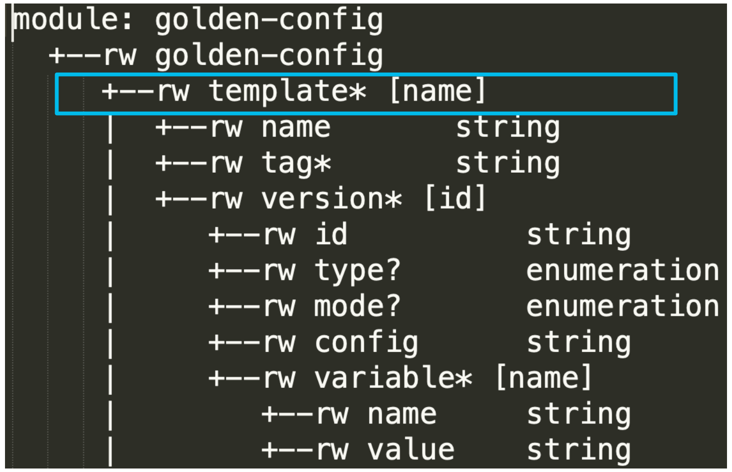
```
router bgp 200  {{ AS }}
vrf testXR  {{ VRF_NAME }}
 rd auto
 bgp router-id 10.20.25.5
 address-family ipv4 unicast
!
neighbor 10.20.25.16  {{ NE }}
remote-as 20{{ AS_REMOTE }}
bfd multiplier   2
ebgp-multihop 5
address-family ipv4 unicast
 allowas-in 5
 site-of-origin 10:4
 route-policy PASS_ALL in
....
```

Supported types:
- Native
- c-style
- j-style
- XML
- JSON

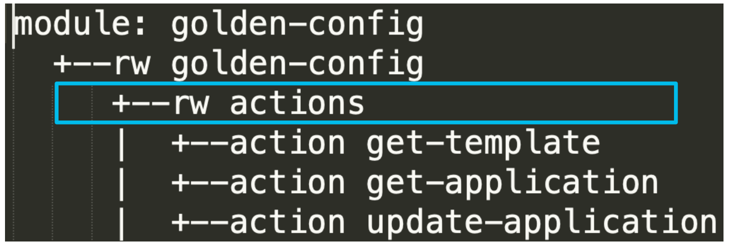




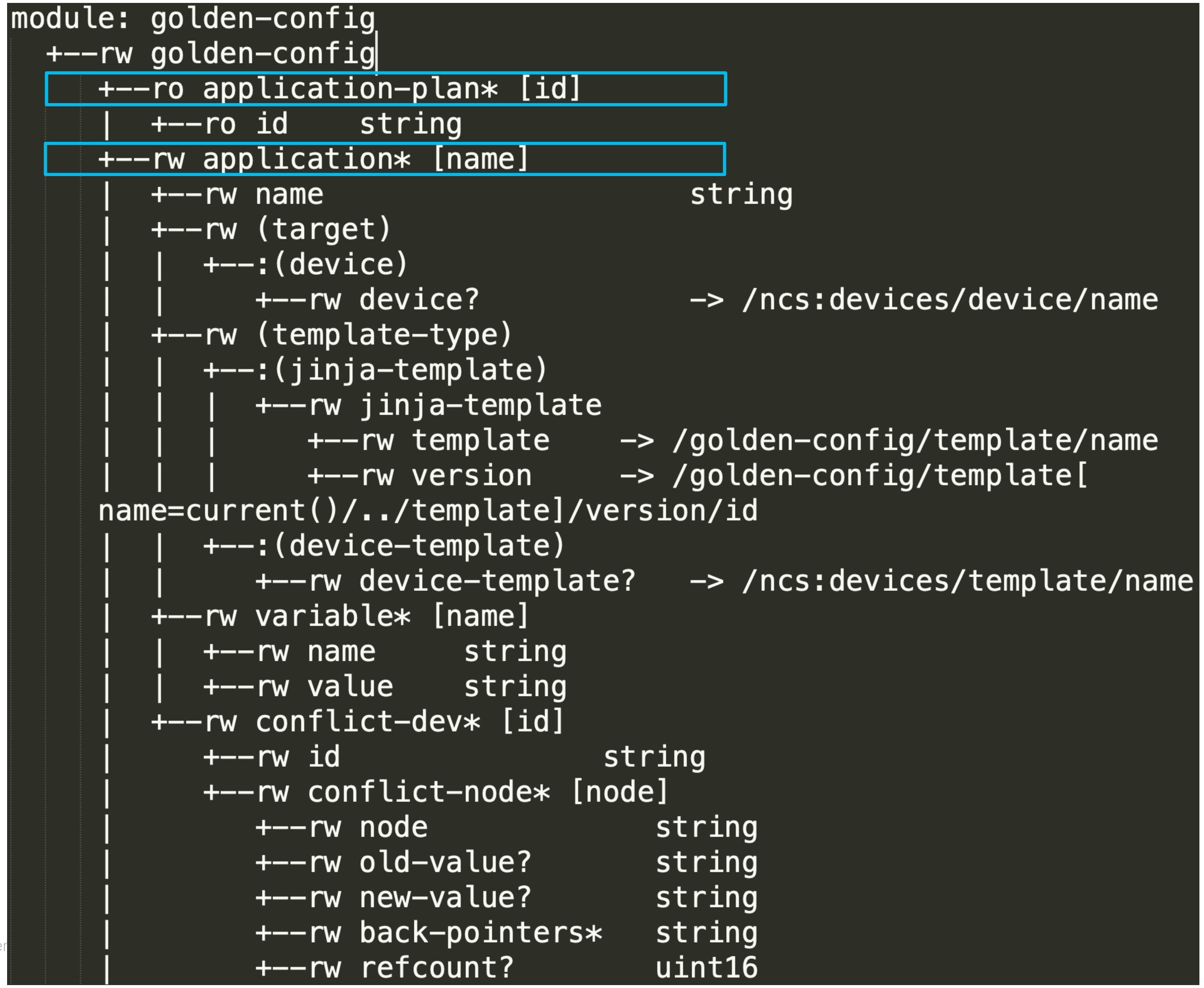
YANG Models



```
module: golden-config
  +--rw golden-config
     +--rw template* [name]
     |  +--rw name          string
     |  +--rw tag*          string
     |  +--rw version* [id]
     |     +--rw id            string
     |     +--rw type?         enumeration
     |     +--rw mode?         enumeration
     |     +--rw config        string
     |     +--rw variable* [name]
     |        +--rw name       string
     |        +--rw value      string
```

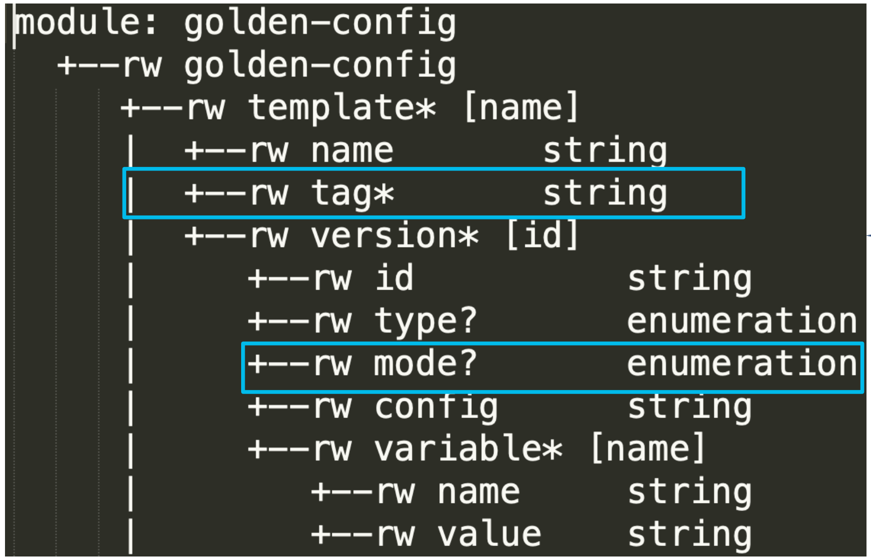


```
module: golden-config
  +--rw golden-config
     +--rw actions
     |  +--action get-template
     |  +--action get-application
     |  +--action update-application
```

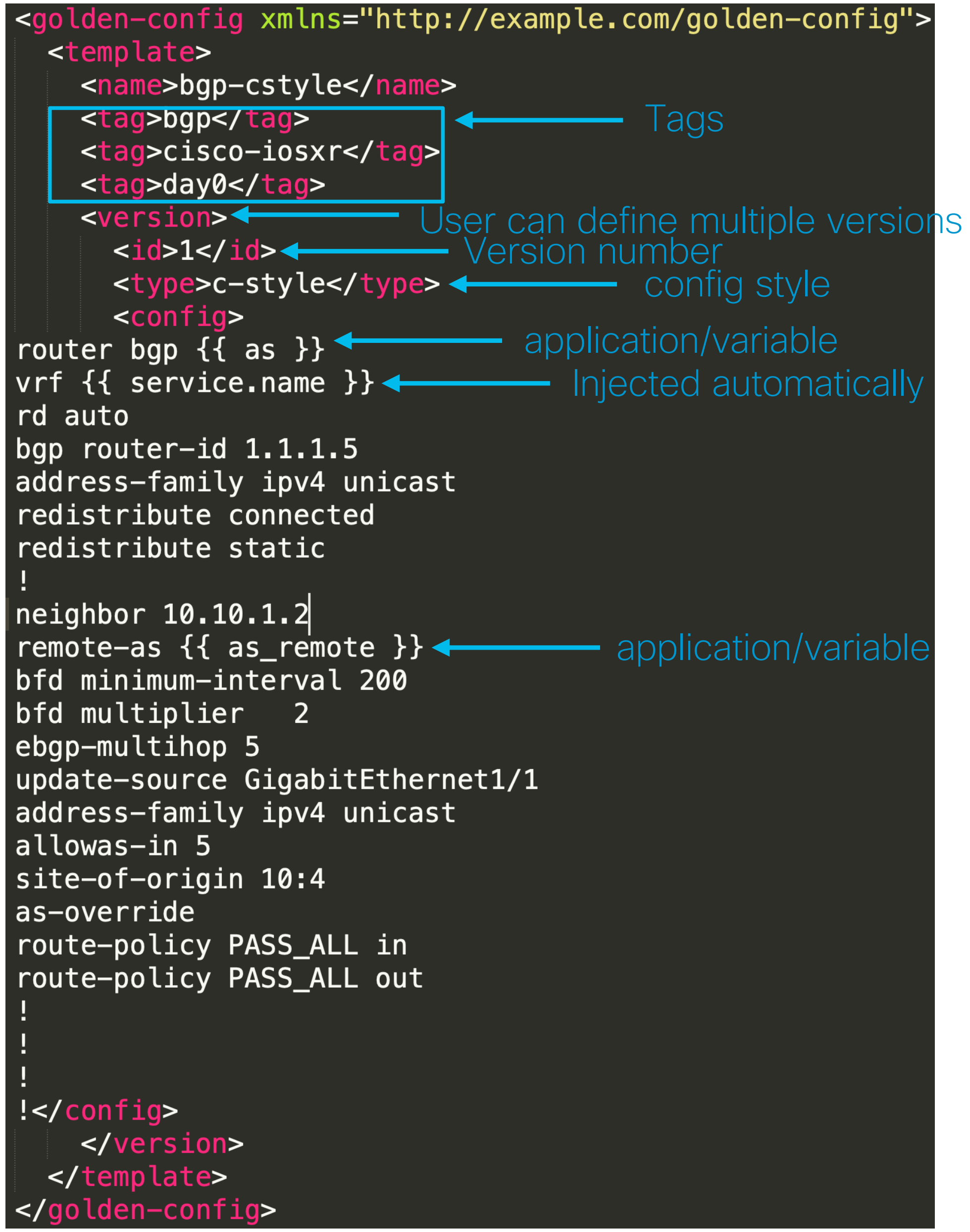


```
module: golden-config
  +--rw golden-config
     +--ro application-plan* [id]
     |  +--ro id     string
     +--rw application* [name]
     |  +--rw name                         string
     |  +--rw (target)
     |  |  +--:(device)
     |  |     +--rw device?                -> /ncs:devices/device/name
     |  +--rw (template-type)
     |  |  +--:(jinja-template)
     |  |  |  +--rw jinja-template
     |  |  |     +--rw template     -> /golden-config/template/name
     |  |  |     +--rw version      -> /golden-config/template[
name=current()/../template]/version/id
     |  |  +--:(device-template)
     |  |     +--rw device-template?   -> /ncs:devices/template/name
     |  +--rw variable* [name]
     |  |  +--rw name     string
     |  |  +--rw value    string
     |  +--rw conflict-dev* [id]
     |     +--rw id                   string
     |     +--rw conflict-node* [node]
     |        +--rw node                 string
     |        +--rw old-value?           string
     |        +--rw new-value?           string
     |        +--rw back-pointers*       string
     |        +--rw refcount?            uint16
```

Example Template: YANG and Payload

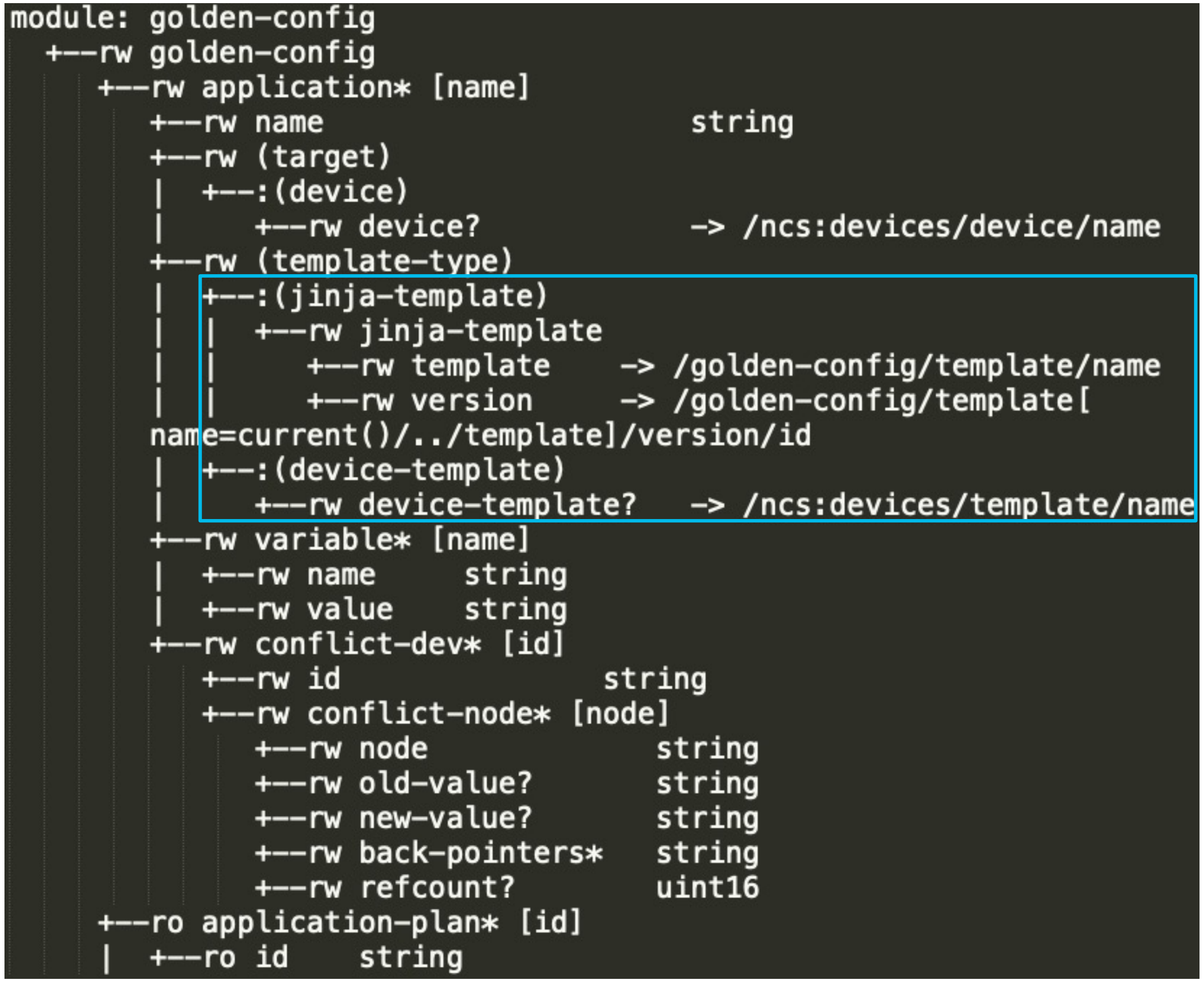


```
module: golden-config
  +--rw golden-config
     +--rw template* [name]
        |  +--rw name           string
        |  +--rw tag*           string
        |  +--rw version* [id]
        |     +--rw id             string
        |     +--rw type?          enumeration
        |     +--rw mode?          enumeration
        |     +--rw config         string
        |     +--rw variable* [name]
        |        +--rw name        string
        |        +--rw value       string
```

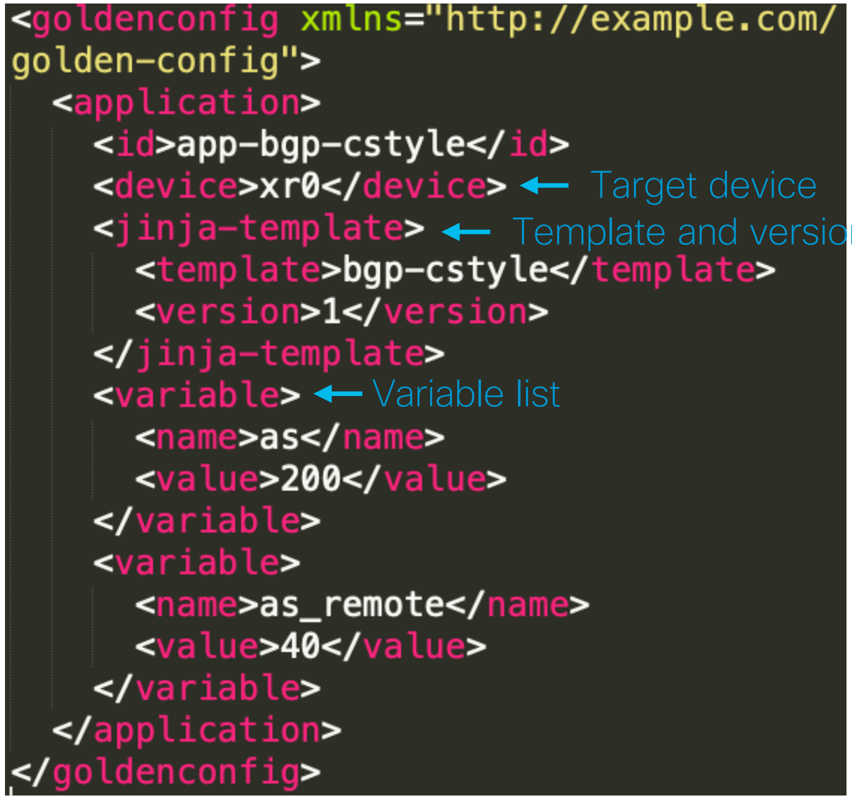


```
<golden-config xmlns="http://example.com/golden-config">
  <template>
    <name>bgp-cstyle</name>
    <tag>bgp</tag>
    <tag>cisco-iosxr</tag>
    <tag>day0</tag>
    <version>
      <id>1</id>
      <type>c-style</type>
      <config>
router bgp {{ as }}
vrf {{ service.name }}
rd auto
bgp router-id 1.1.1.5
address-family ipv4 unicast
redistribute connected
redistribute static
!
neighbor 10.10.1.2
remote-as {{ as_remote }}
bfd minimum-interval 200
bfd multiplier   2
ebgp-multihop 5
update-source GigabitEthernet1/1
address-family ipv4 unicast
allowas-in 5
site-of-origin 10:4
as-override
route-policy PASS_ALL in
route-policy PASS_ALL out
!
!
!
!</config>
    </version>
  </template>
</golden-config>
```

Example Application: YANG and Payload



```
module: golden-config
  +--rw golden-config
     +--rw application* [name]
        +--rw name                        string
        +--rw (target)
        |  +--:(device)
        |     +--rw device?               -> /ncs:devices/device/name
        +--rw (template-type)
        |  +--:(jinja-template)
        |  |  +--rw jinja-template
        |  |     +--rw template     -> /golden-config/template/name
        |  |     +--rw version      -> /golden-config/template[
        name=current()/../template]/version/id
        |  +--:(device-template)
        |     +--rw device-template?   -> /ncs:devices/template/name
        +--rw variable* [name]
        |  +--rw name      string
        |  +--rw value     string
        +--rw conflict-dev* [id]
           +--rw id                  string
           +--rw conflict-node* [node]
              +--rw node               string
              +--rw old-value?         string
              +--rw new-value?         string
              +--rw back-pointers*     string
              +--rw refcount?          uint16
     +--ro application-plan* [id]
     |  +--ro id     string
```



```
<goldenconfig xmlns="http://example.com/
golden-config">
  <application>
    <id>app-bgp-cstyle</id>
    <device>xr0</device>
    <jinja-template>
      <template>bgp-cstyle</template>
      <version>1</version>
    </jinja-template>
    <variable>
      <name>as</name>
      <value>200</value>
    </variable>
    <variable>
      <name>as_remote</name>
      <value>40</value>
    </variable>
  </application>
</goldenconfig>
```

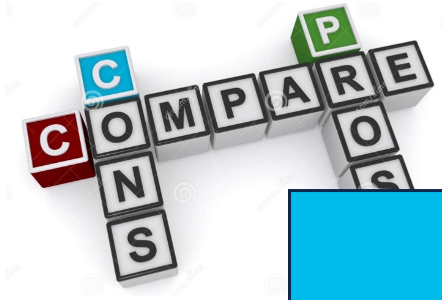
Use of Jinja2

Jinja > NSO

Widely used Template engine for Network Automation

- Render your first network configuration template using Python and Jinja2
- Generate Cisco Configuration Template Using Python3, Jinja2
- Generate Cisco Layer2 Switch Config from Port Management Table and Jinja2 Template
- Jinja2 Tutorial - Loops and Conditions

many more....



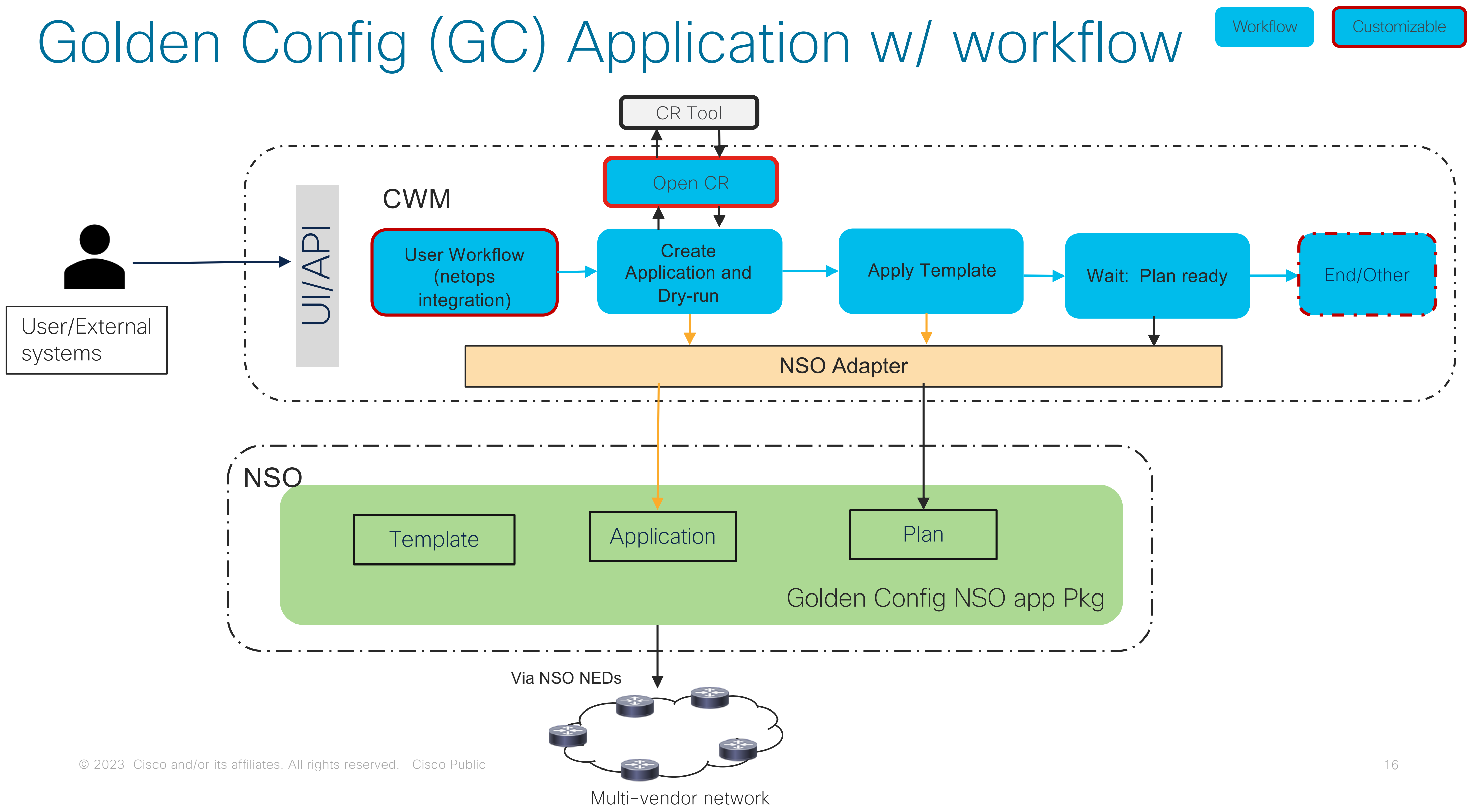
Templates Comparison

Comparison Point	Service Template	Device Template	GC Template
Independent of use case specific NSO Package	✗	✓	✓
Device native config	✗	✗	✓
No knowledge of service/ned YANG model	✗	✗	✓
Supports Dry-run	✓	✓	✓
FastMap Features • Free Deletes • Propagate Updates/Easy Remediation • Reconciliation • Retry etc.	✓	✗	✓
Template tagging – group operations	✗	✗	✓
Programming capabilities	✓	✗	✓

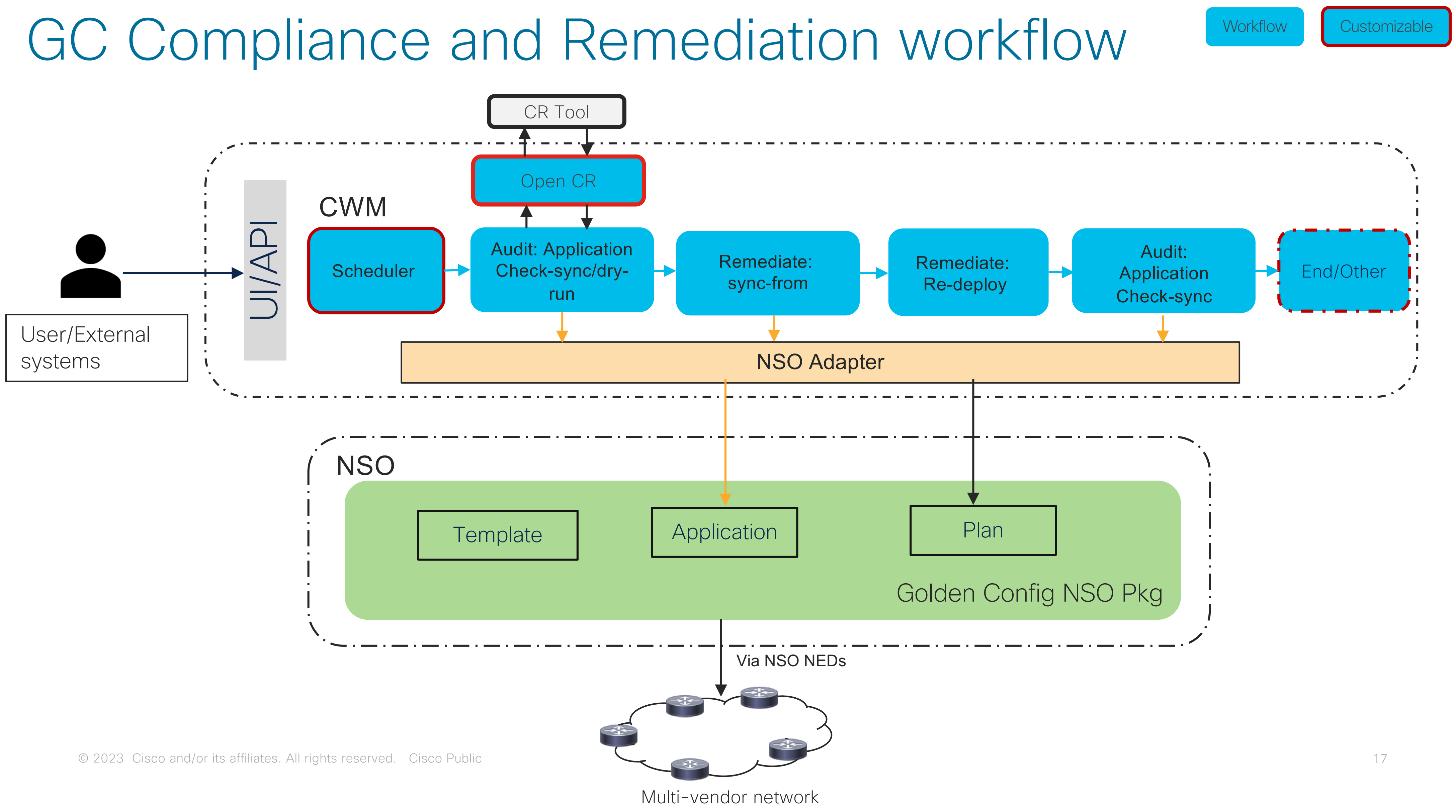
FastMap + Template Engine = Golden Config Templates

Together with Crosswork Workflow Manager

Golden Config (GC) Application w/ workflow



GC Compliance and Remediation workflow

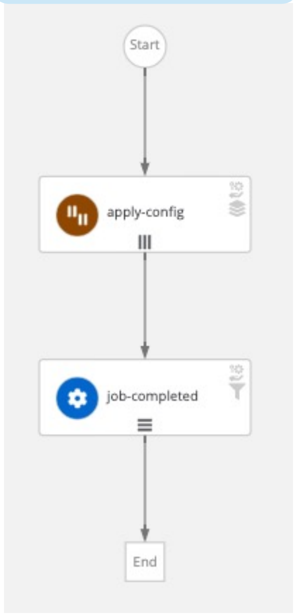


Golden Config

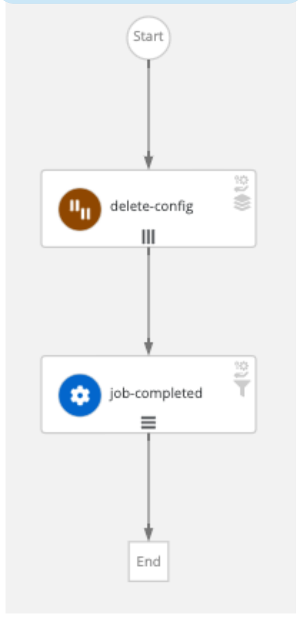
With Crosswork Workflow Manager

- Golden config application and actions to be instantiated/run from CWM

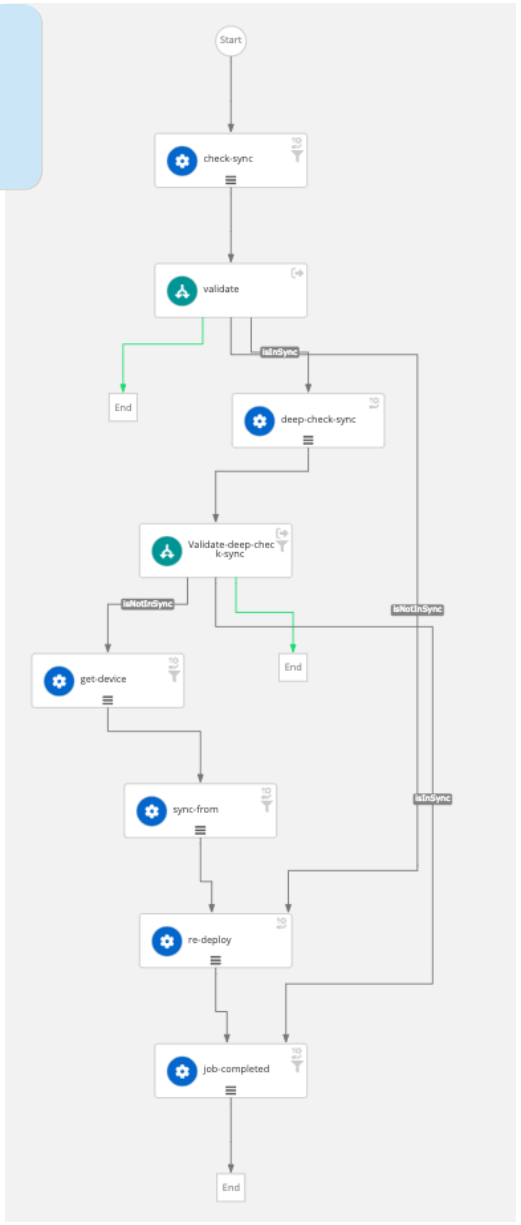
Creation Workflow



Deletion Workflow



Remediation Workflow



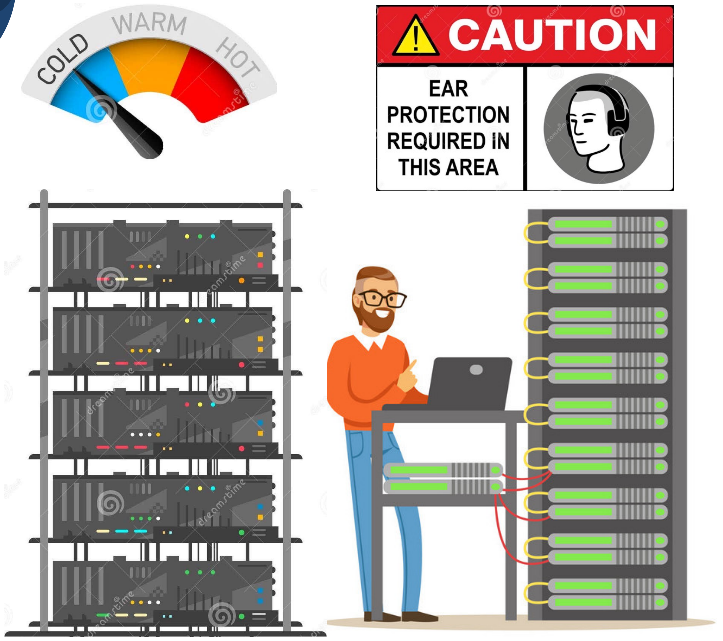
Demo

- Golden Config Templates through NSO

Demo

- Workflows for working with Golden Config Templates

Questions



Join us for Lab

Tomorrow 11am – 1pm

The bridge to possible