Exploring YANG Data Modeling

A Guided Session using NSO

Aayushi Mahajan
Software Consulting Engineer

CISCO Live !

Cisco Webex App

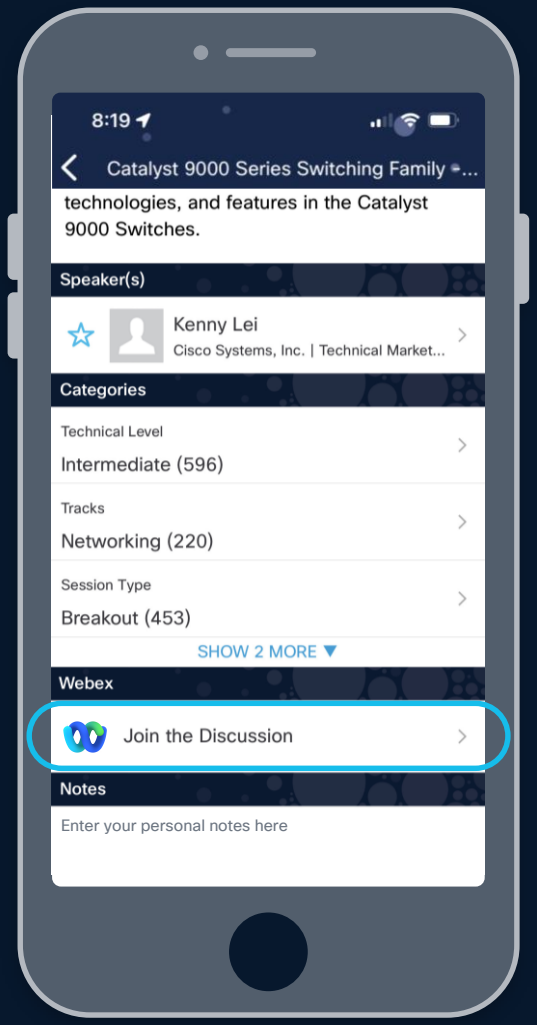
Questions?

Use Cisco Webex App to chat
with the speaker after the session

How

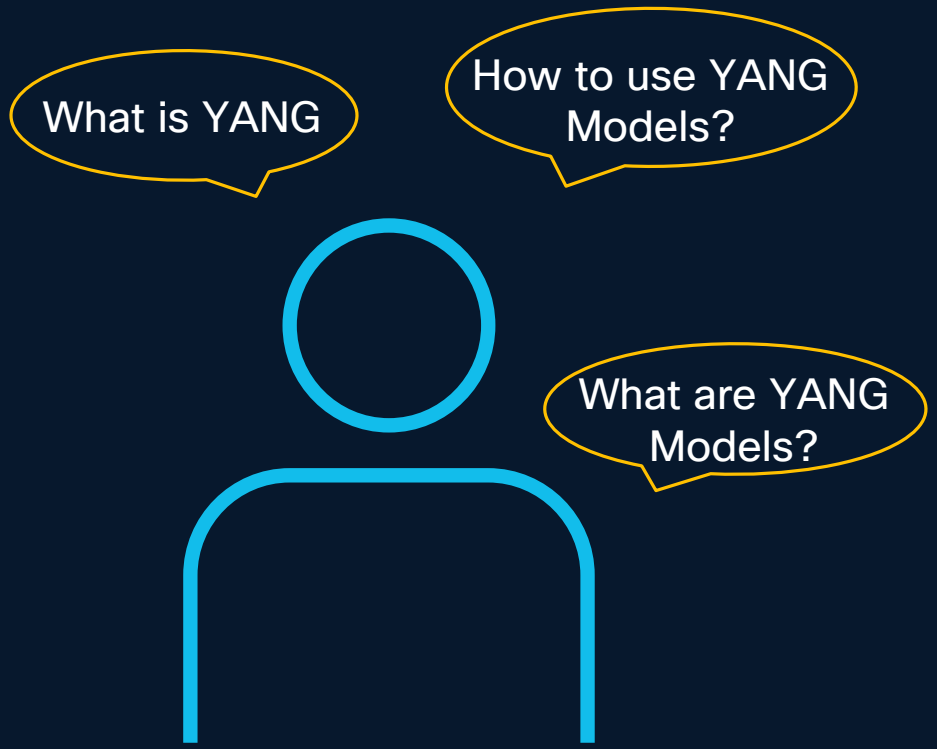
1. Find this session in the Cisco Live Mobile App
2. Click “Join the Discussion”
3. Install the Webex App or go directly to the Webex space
4. Enter messages/questions in the Webex space

Webex spaces will be moderated by the speaker until June 13, 2025.



https://ciscolive.ciscoevents.com/
ciscolivebot/#DEVNET-1882

Agenda



01 Need for Data Modeling

02 What is YANG

03 Why YANG?

04 Components & Data Types in YANG

05 Cisco NSO Playground overview

06 Live Demo

07 Recap & Takeaways

08 Slido Quiz

09 Q&A

10 Appendix

Need for Data Modeling

Unstructured Data: Not Always Easy to Read

```
Router Re1-BGL-12-3 1.2.3.4 down GigabitEthernet 0/0/2 enable
```

What do these commands mean?

```
person Jacob 27 hawai vegan
```

How would a machine know the datatypes and relationships between them?

Structured Data

```
<config xmlns="http://tail-f.com/ns/config/1.0">
  <person xmlns="http://com/example/userinfo">
    <name>Aayushi</name>
    <age>26</age>
    <favorite-holiday-destinations>San Diego</favorite-holiday-destinations>
    <food-choice>vegetarian</food-choice>
  </person>
</config>
```

Example 1

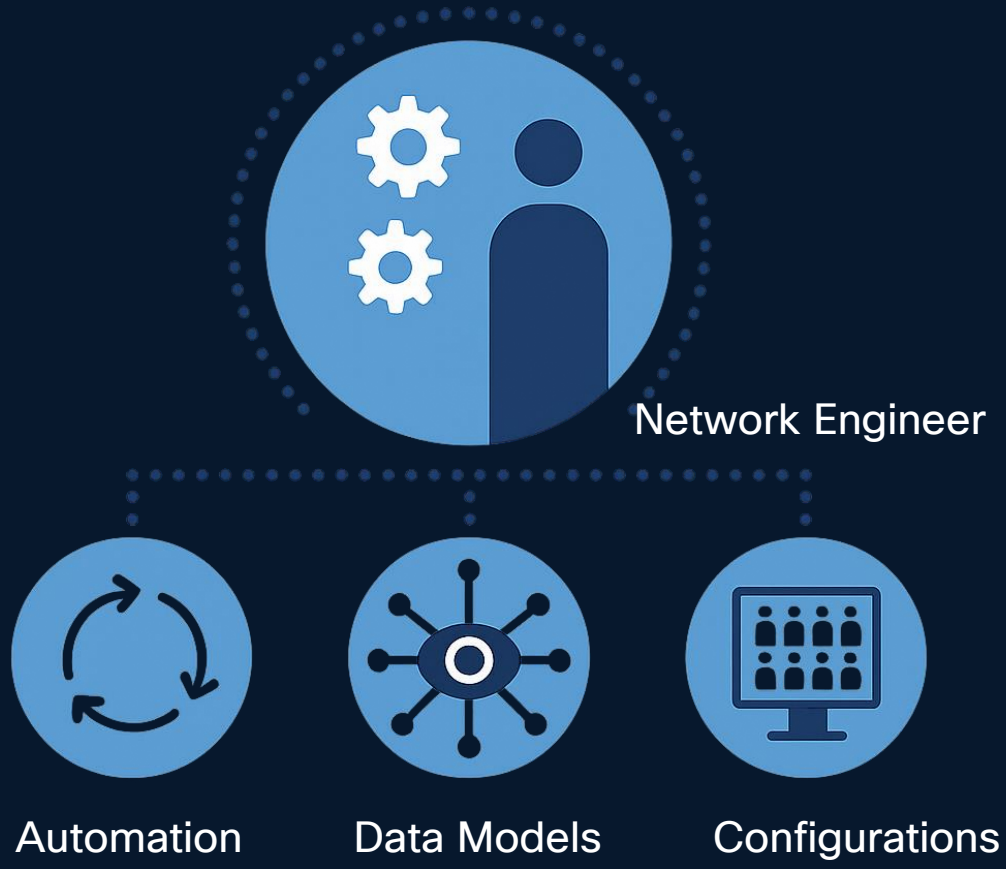
Example 2

```
<config xmlns="http://tail-f.com/ns/config/1.0">
  <router xmlns="http://com/example/routermodel">
    <name>rno1-BGL-16-1</name>
    <address>10.20.30.40</address>
    <operational-status>up</operational-status>
    <interface>
      <type>GigabitEthernet</type>
      <enabled>true</enabled>
      <interface-id>1/1</interface-id>
    </interface>
  </router>
</config>
```

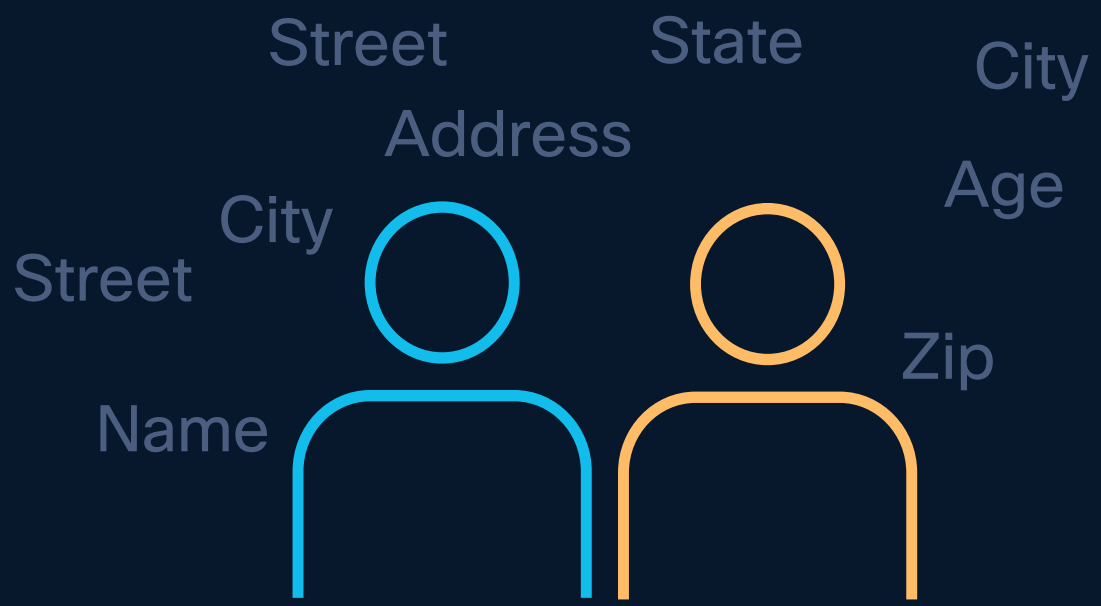
Who defines the STRUCTURE?

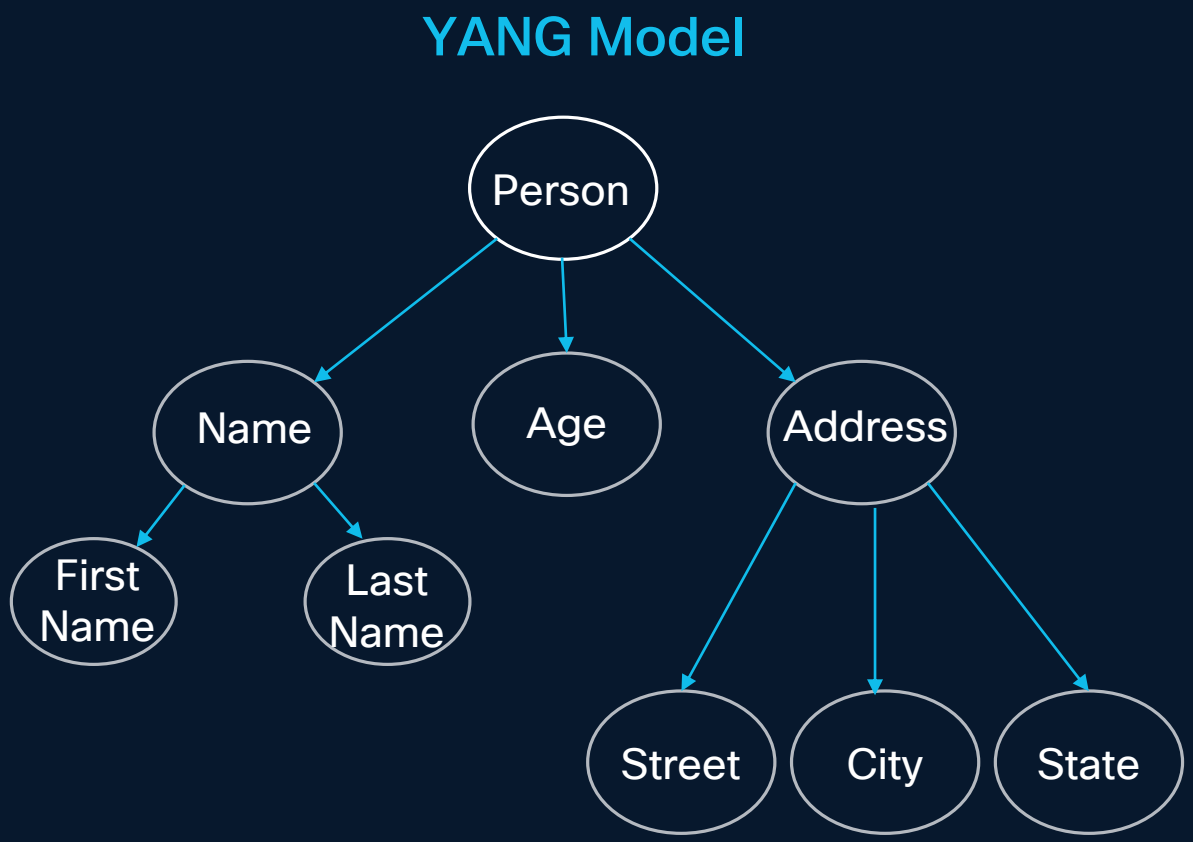
YANG

Yet Another Next Generation



Blueprint for network configuration





Why YANG?

Yang is the key to automating, scaling, and future-proofing your network

AI generated image

Features of YANG

1. Yang is a tree hierarchy
   - Starts at the root (top)
2. Defines the structure
3. Modularity
4. YANG supports different encoding formats (e.g. XML, JSON).
5. Helps automation tooling know:
   - What features are available
   - Expected Input/Output data constraints

Criteria	YANG	XML Schema / JSON Schema
Designed for networking	✅ Purpose-built for modeling config/state data in network devices	❌ Generic structure validation, not network-aware
Human-readable & concise	✅ Tree-structured, easy to read and maintain	⚠️ Verbose (XML), less intuitive
Standardized by IETF	✅ Widely adopted in RFCs (RFC 6020/7950)	❌ Not designed for network standards
Supports modularity & reuse	✅ Uses import, include, grouping	⚠️ Limited or verbose reuse
Integration with protocols	✅ Works seamlessly with NETCONF, RESTCONF	❌ Doesn't integrate directly

XML Schema

```
<config xmlns="http://tail-f.com/ns/config/1.0">
  <router xmlns="http://com/example/routermodel">
    <name>rno1-BGL-16-1</name>
    <address>10.20.30.40</address>
    <operational-status>up</operational-status>
    <interface>
      <type>GigabitEthernet</type>
      <enabled>true</enabled>
      <interface-id>1/1</interface-id>
    </interface>
  </router>
</config>
```

YANG Model

```
container router {
  leaf name {
    type string;
  }
  leaf address {
    type inet:ipv4-address;
  }
  leaf operational-status {
    type enumeration {
      enum up;
      enum down;
    }
  }
  list interface {
    key "type";

    description
      "The list of interfaces on the device.";

    leaf type {
      type enumeration {
        enum GigabitEthernet;
        enum TenGigE;
      }
    }
    leaf enabled {
      type boolean;
      default "true";
      description "This leaf contains the configured, desired state of the interface.";
    }
    leaf interface-id {
      tailf:info "X.X or X/X";
      type union {
        type string{
          pattern '(\d+)(.)(\d+)';
        }
        type string {
          pattern '(\d+/)+\d+';
        }
      }
```

```
<config xmlns="http://tail-f.com/ns/config/1.0">
  <router xmlns="http://com/example/routermodel">
    <name>rno1-BGL-16-1</name>
    <address>10.20.30.40</address>
    <operational-status>up</operational-status>
    <interface>
      <type>GigabitEthernet</type>
      <enabled>true</enabled>
      <interface-id>1/1</interface-id>
    </interface>
  </router>
</config>
```

Components & Data Types in YANG

Components of YANG – Modules, Namespaces & Imports

Module:

- Defines data model structure
- Identified by a unique namespace URI
- Represents a single data model

```
module router-model {
  namespace "http://com/example/routermodel";
  prefix router-model;
  import ietf-inet-types {
    prefix inet;
  }
```

Namespace:

- Uniquely identifies a YANG module
- Prevents naming conflicts across modules
- Typically written as a URI (e.g., http://example.com/ns/module)

```
module router-model {
  namespace "http://com/example/routermodel";
  prefix router-model;
  import ietf-inet-types {
    prefix inet;
  }
```

Import Statement:

- Allows modular design & code reuse
- Uses prefix to refer to imported items
- Enables integration of external types and groupings

```
module router-model {
  namespace "http://com/example/routermodel";
  prefix router-model;
  import ietf-inet-types {
    prefix inet;
  }
```

The Basic Building Blocks of YANG: Node Types

```
container person {
  leaf name {
    type string;
    }
  leaf age {
    type uint32;
  }
  leaf-list favorite-holiday-destinations {
    type string;
  }
  leaf food-choice {
    type enumeration {
      enum vegan;
      enum vegetarian;
      enum non-vegetarian;
    }
    mandatory true;
  }
}
```

CONTAINER

Group nodes together (no data)

LEAF

A single unit of data

LEAF-LIST

Multiple units of one data

LIST

Collection of nodes (leaves & leaf-lists)

```
container router {
  leaf name {
    type string;
  }
  leaf address {
    type inet:ipv4-address;
  }
  leaf operational-status {
    type enumeration {
      enum up;
      enum down;
    }
  }
  list interface {
    key "type";

    description
      "The list of interfaces on the device.";

    leaf type {
      type enumeration {
        enum GigabitEthernet;
        enum TenGigE;
      }
    }
    leaf enabled {
      type boolean;
      default "true";
      description "This leaf contains the configured, desired state of the interface.";
    }
    leaf interface-id {
      tailf:info "X.X or X/X";
      type union {
        type string{
          pattern '(\d+)(.)(\d+)';
        }
        type string {
          pattern '(\d+/)+\d+';
```

Some Basic YANG Data Types

Type	Description
int8/16/32/64	Integer
uint8/16/32/64	Unsigned integer
decimal64	Decimal floating-point format
string	Unicode string
enumeration	Set of alternatives
boolean	True or false
bits	Boolean array
binary	Binary BLOB
leafref	Reference "pointer"

```
container router {
  leaf name {
    type string;
  }
  leaf address {
    type inet:ipv4-address;
  }
  leaf operational-status {
    type enumeration {
      enum up;
      enum down;
    }
  }
  list interface {
    key "type";

    description
      "The list of interfaces on the device.";

    leaf type {
      type enumeration {
        enum GigabitEthernet;
        enum TenGigE;
      }
    }
    leaf enabled {
      type boolean;
      default "true";
      description "This leaf contains the configured, desired state of the interface.";
    }
    leaf interface-id {
      tailf:info "X.X or X/X";
      type union {
        type string{
          pattern '(\d+)(.)(\d+)';
        }
        type string {
          pattern '(\d+/)+\d+';
        }
      }
```

String: A flexible mix of numbers, letters & punctation

Custom Imported Types:

- Hidden regular expressions enforcing ipv4 syntax

Enumeration:

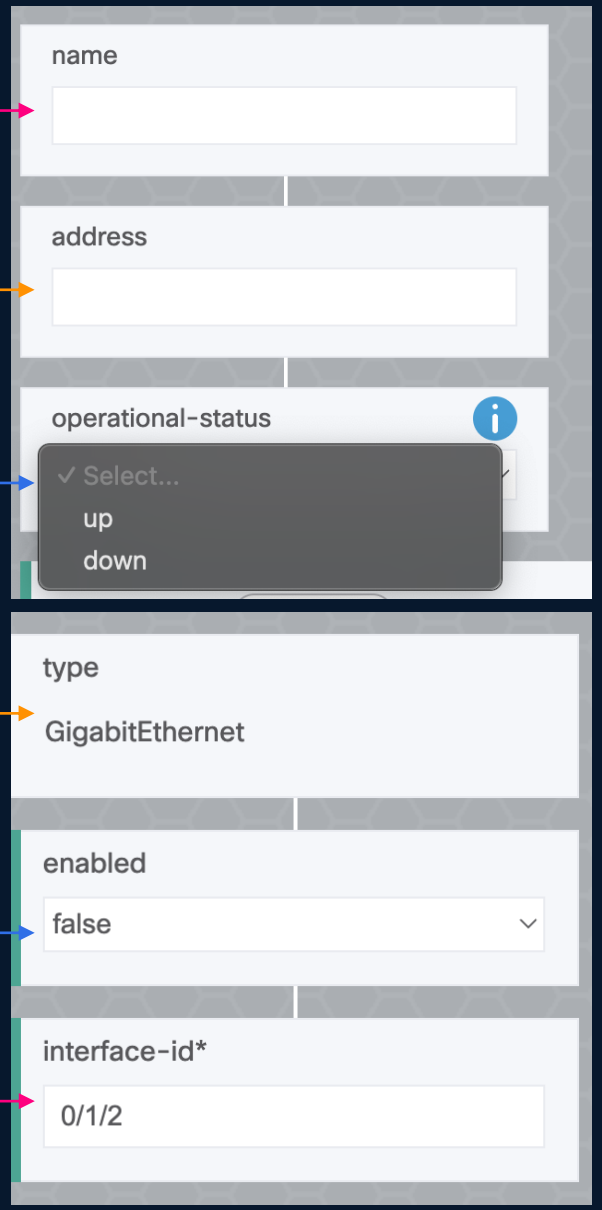
- Choose one from pre-defined set of value
- Has additional enum for each value

Boolean: Whether a value is either True or False

Union: The value should correspond to one of its member types

YANG Statements Visualized: NSO GUI

```
container router {
  leaf name {
    type string;
  }
  leaf address {
    type inet:ipv4-address;
  }
  leaf operational-status {
    type enumeration {
      enum up;
      enum down;
    }
  }
  list interface {
    key "type";

    description
      "The list of interfaces on the device.";

    leaf type {
      type enumeration {
        enum GigabitEthernet;
        enum TenGigE;
      }
    }
    leaf enabled {
      type boolean;
      default "true";
      description "This leaf contains the configured, desired state of the interface.";
    }
    leaf interface-id {
      tailf:info "X.X or X/X";
      type union {
        type string{
          pattern '(\d+)(.)(\d+)';
        }
        type string {
          pattern '(\d+/)+\d+';
        }
      }
```



cisco

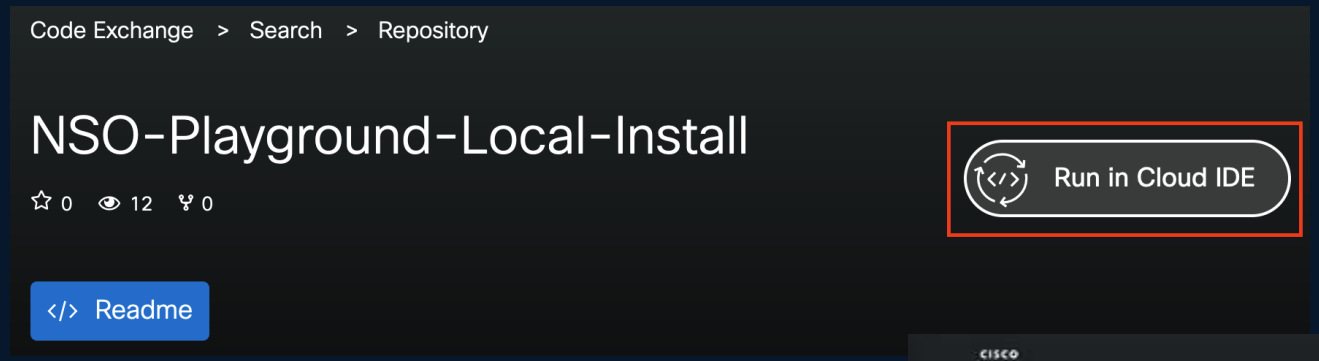
YANG Statements Visualized: NSO CLI

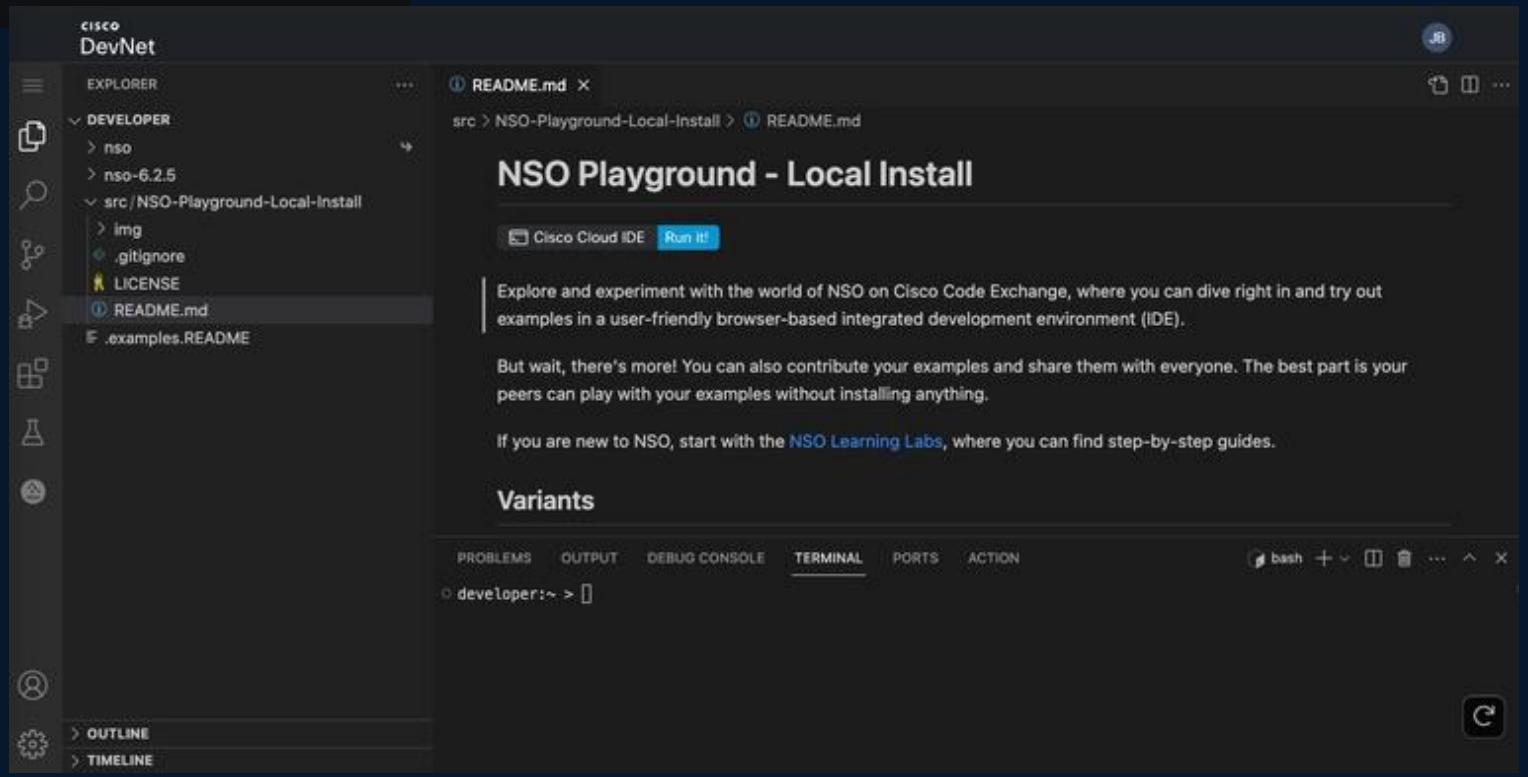
```
container router {
  leaf name {
    type string;
  }
  leaf address {
    type inet:ipv4-address;
  }
  leaf operational-status {
    type enumeration {
      enum up;
      enum down;
    }
  }
  list interface {
    key "type";

    description
      "The list of interfaces on the device.";

    leaf type {
      type enumeration {
        enum GigabitEthernet;
        enum TenGigE;
      }
    }
    leaf enabled {
      type boolean;
      default "true";
      description "This leaf contains the configured, desired state of the interface.";
    }
    leaf interface-id {
      tailf:info "X.X or X/X";
      type union {
        type string{
          pattern '(\d+)(.)(\d+)';
        }
        type string {
          pattern '(\d+/)+\d+';
        }
      }
```

```
admin@ncs(config)# router ?
Possible completions:
  address  interface  name  operational-status
admin@ncs(config)# router operational-status ?
Possible completions:
  down  up
admin@ncs(config)# router interface
Possible completions:
  GigabitEthernet  TenGigE
admin@ncs(config)# router interface TenGigE ?
Possible completions:
  disabled
  enabled
  interface-id   X.X or X/X
  <cr>
```

Cisco NSO Playground Overview

Getting started with NSO Playground





Creating an NSO Instance

Step 1: Prepare NSO

```
source $NCS_DIR/ncsrc
```

Step 2: Setup your NSO instance

```
ncs-setup --dest ~/nso-instance
```

Step 3: Go to the ~/nso-instance directory

```
cd ~/nso-instance
```

Step 4: Start the NSO server

```
ncs
```

Compile / Install YANG Packages

Step 1: Prepare NSO

```
source $NCS_DIR/ncsrc
```

Step 2: Copy the package directories into ~/nso-instance/packages by:

- Cloning the packages using git clone into your home directory, then moving or soft-linking them into the packages
- Dragging and dropping the folders via the VS Code GUI

```
git clone https://github.com/aaymahaj/DEVNET-1882

cd ~/nso-instance/packages

# Move the directories from DEVNET-1882 to the packages directory
mv ~/DEVNET-1882/user-info ~/nso-instance/packages/
mv ~/DEVNET-1882/router-model ~/nso-instance/packages/
```

Step 3: Re-compile the packages

```
cd ~/nso-instance/packages/user-info/src
make clean all
cd ~/nso-instance/packages/router-model/src
make clean all
```

Step 4: Reload the packages

```
packages reload force
exit
```

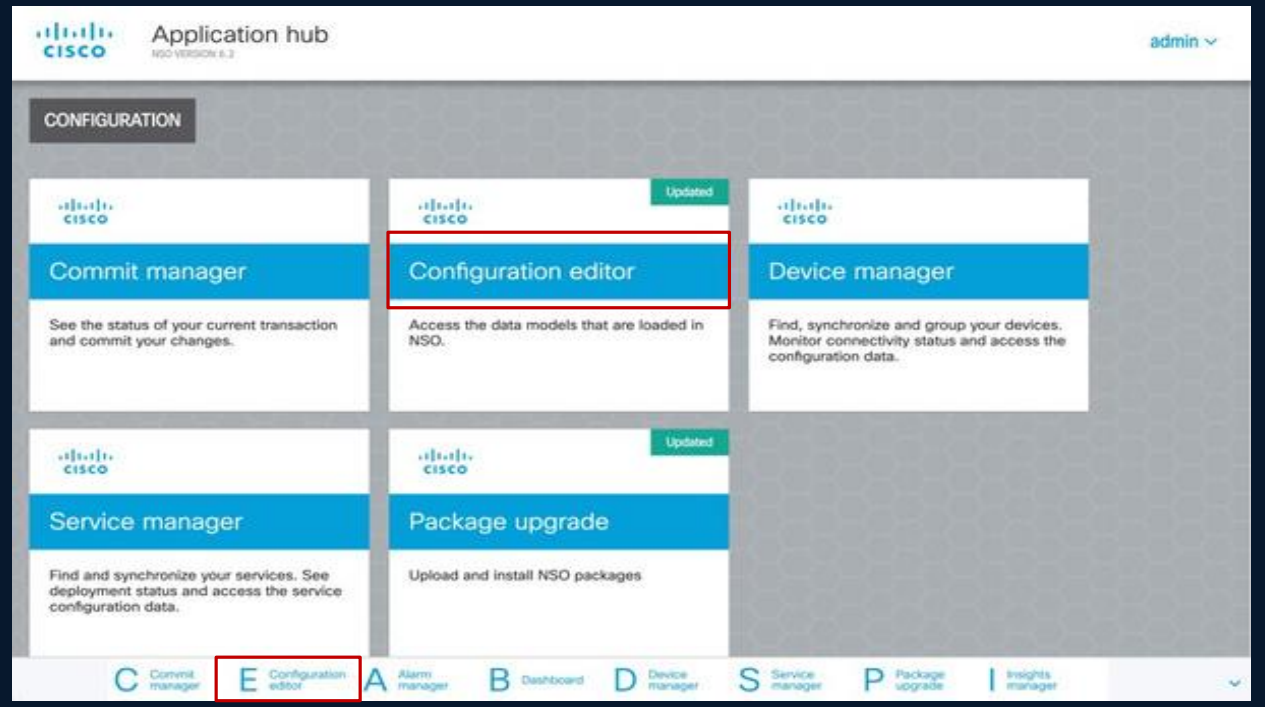
Accessing the NSO GUI

Step 1: echo NSO GUI URL

```
echo $DEVENV_APP_8080_URL
```

Step 2: Open the above URL in a browser
The credentials are ”admin” / “admin”

Step 3: Click on the “E – Configuration Editor” on the bottom left
(Or on the button in the middle of the screen}

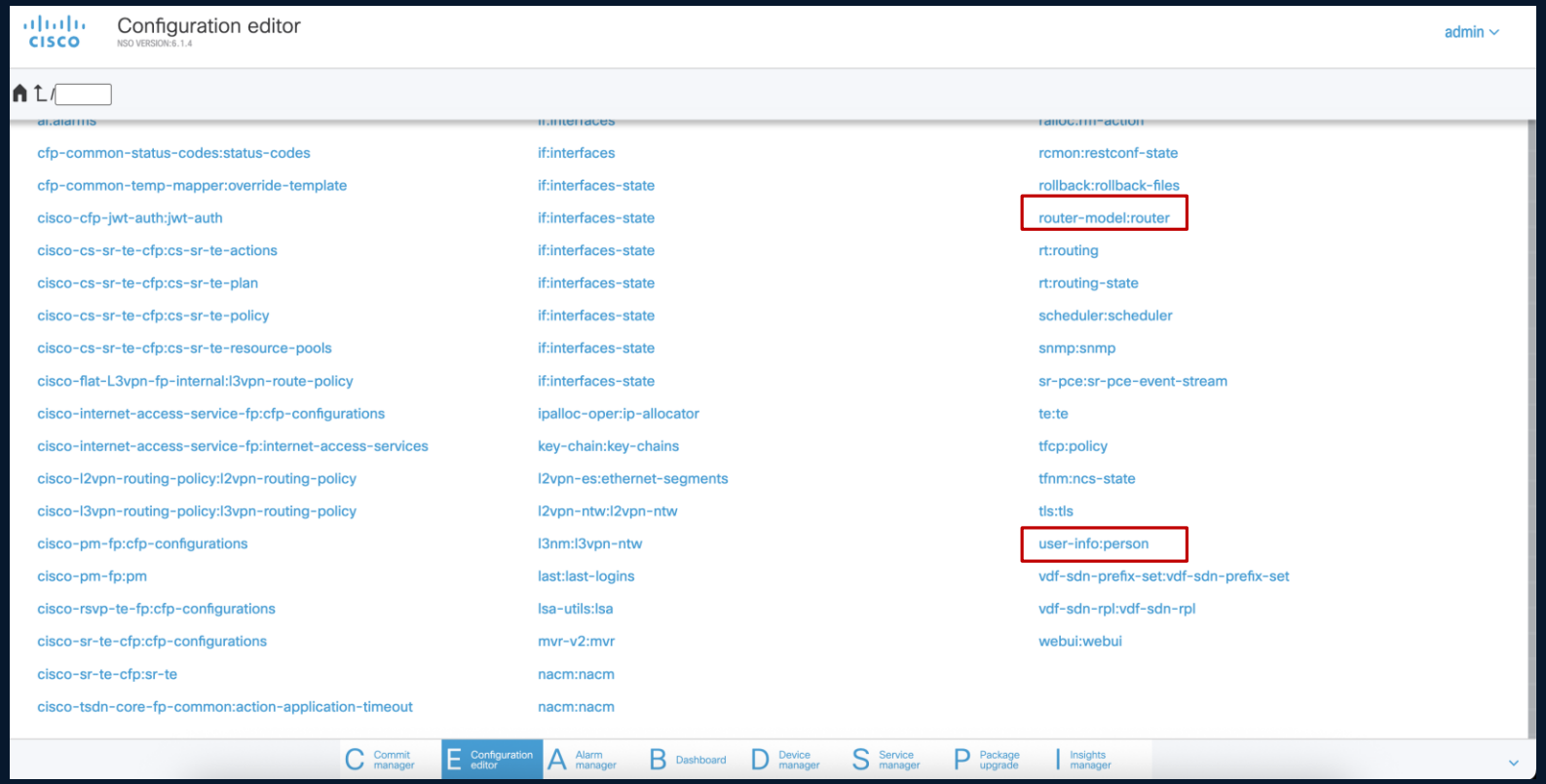


Accessing the Packages on GUI

Step 1: Each package is assigned a unique namespace prefix in the format:

<namespace>:<package_name>

Step 2: Select a custom package to edit its configurations.
By default, no values are pre-populated—configurations start as blank.

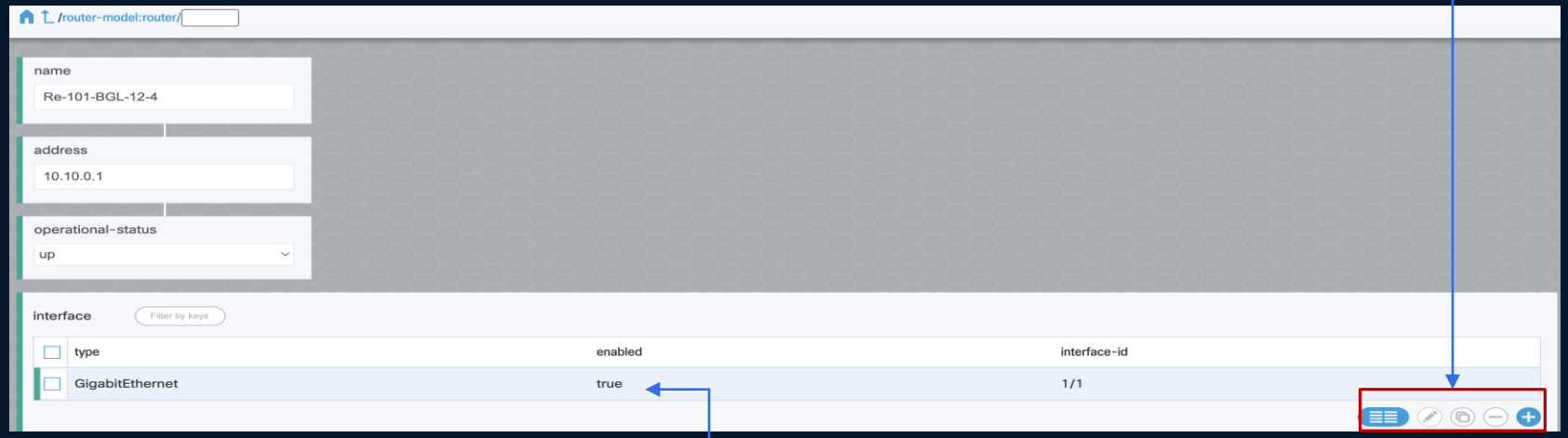


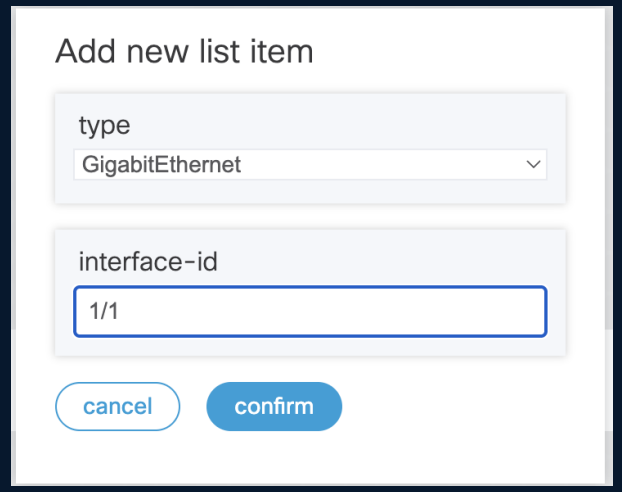
Adding Configurations to our custom fields via GUI

Step 1: Enter a values in the respective fields.

Step 2: Provide a valid IP address in the “address” field.

Step 3: Click on “+” to add list items in list “Interface”





Step 4: The value for “enabled” is “True” by default and can be updated by clicking on it.

Note: The data model uses a custom data type for IP address validation. A regular expression (RegEx) ensures that only properly formatted IP addresses are accepted.

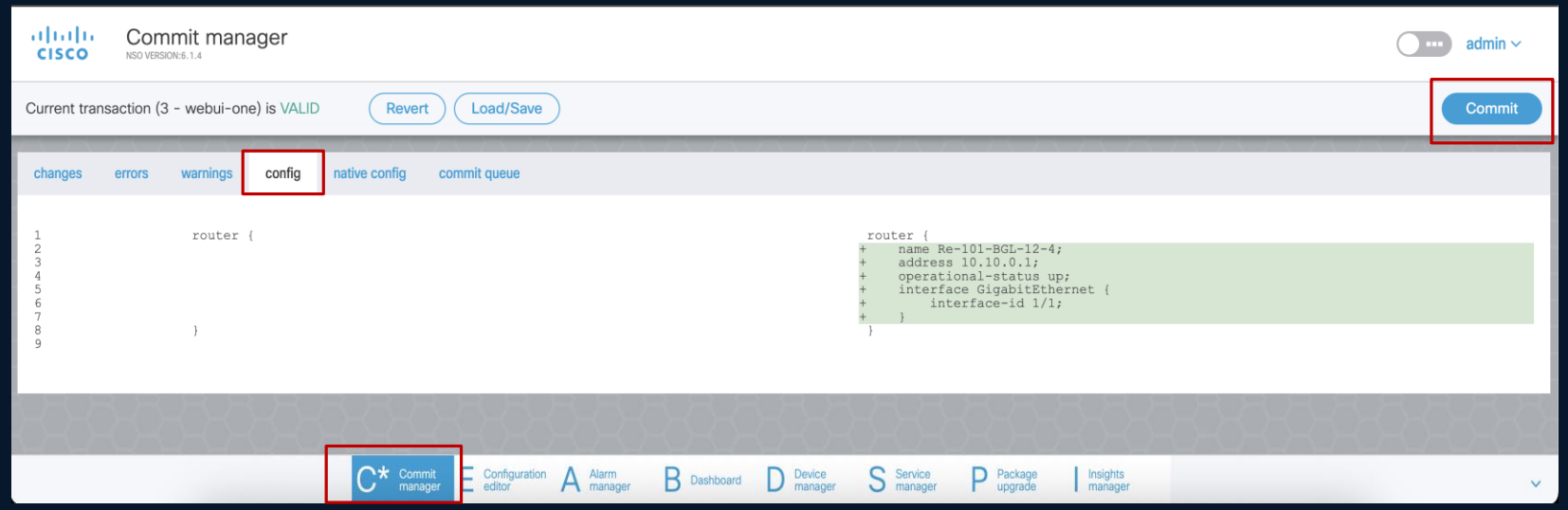
Saving the Data (commit) to NSO

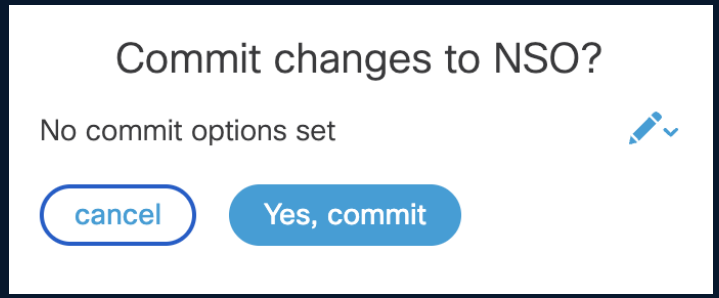
Step 1: Click on the “C – Commit Manager” on the bottom left

Step 2: Make sure you have entered the right values by clicking on “Config” to see the dry-run

Step 3: Click on “Commit” button

Step 4: Click on “Yes, commit” button

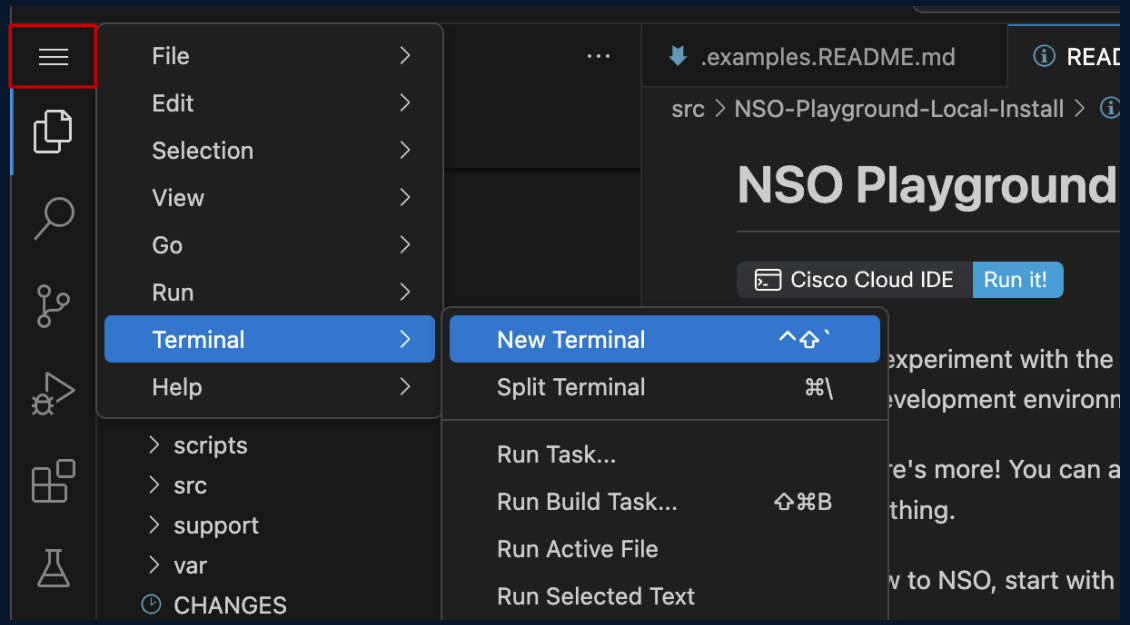




Accessing the NSO CLI from Terminal in the Browser

Step 1: If a terminal is already visible at the bottom of the screen, use it directly.

Step 2: If not, open a new terminal by clicking on the **Terminal** icon in the left-side navigation bar.



Adding Configurations to our custom fields via CLI

Step 1: Prepare NSO

```
source $NCS_DIR/ncsrc
```

Step 2: Enter in NSO CLI

```
ncs_cli -C -u admin
```

Step 3: To view the configurations, we committed in the cli, run the below commands

```
show running-config router
show running-config router | display json
show running-config router | display xml
```

Step 4: Enter "**config**" mode to start configuring, follow these steps as shown in the adjacent image

```
admin@ncs# config
Entering configuration mode terminal
admin@ncs(config)# router name re-demo-CLUS address 4.5.6.7 ?
Possible completions:
  interface  operational-status  <cr>
admin@ncs(config)# router name re-demo-CLUS address 4.5.6.7 operational-status up interface ?
Possible completions:
  GigabitEthernet  TenGigE
admin@ncs(config)# router name re-demo-CLUS address 4.5.6.7 operational-status up interface TenGigE ?
Possible completions:
  disabled
  enabled
  interface-id   X.X or X/X
  <cr>
admin@ncs(config)# router name re-demo-CLUS address 4.5.6.7 operational-status up interface TenGigE interface-id 1.4 ?
Possible completions:
  disabled  enabled  <cr>
admin@ncs(config)# router name re-demo-CLUS address 4.5.6.7 operational-status up interface TenGigE interface-id 1.4 disabled
admin@ncs(config-interface-TenGigE)# commit
Commit complete.
admin@ncs(config)# show full-configuration router
router name        re-demo-CLUS
router address     4.5.6.7
router operational-status up
router interface TenGigE
 disabled
 interface-id 1.4
!
```

View Configurations in CLI

Step 1: Prepare NSO

```
source $NCS_DIR/ncsrc
```

Step 2: Enter in nso cli

```
ncs_cli -C -u admin
```

Step 3: To view the configurations, we committed in the cli, run the below commands

```
show running-config router
show running-config router | display json
show running-config router | display xml
```

output

```
admin@ncs# show running-config router
router name        Re-101-BGL-12-4
router address     10.10.0.1
router operational-status up
router interface GigabitEthernet
 interface-id 1/1
!
admin@ncs# show running-config router | display json
{
  "data": {
    "router-model:router": {
      "name": "Re-101-BGL-12-4",
      "address": "10.10.0.1",
      "operational-status": "up",
      "interface": [
        {
          "type": "GigabitEthernet",
          "interface-id": "1/1"
        }
      ]
    }
  }
}
admin@ncs# show running-config router | display xml
<config xmlns="http://tail-f.com/ns/config/1.0">
  <router xmlns="http://com/example/routermodel">
    <name>Re-101-BGL-12-4</name>
    <address>10.10.0.1</address>
    <operational-status>up</operational-status>
    <interface>
      <type>GigabitEthernet</type>
      <interface-id>1/1</interface-id>
    </interface>
  </router>
</config>
```

Live Demo

Recap & Takeaways

Recap & Takeaways

What You Learned:

- Why structured data modeling is essential in networking
- The fundamentals and components of YANG
- How YANG fits into the Cisco NSO ecosystem
- How to visualize and deploy YANG models using NSO (GUI + CLI)

Key Takeaways:

- YANG simplifies data structure definitions for automation
- NSO + YANG enables scalable, modular network config management
- Start small—experiment with basic modules in the NSO Playground
- Use YANG models to bridge human readability and machine automation

Slido Quiz

Join at slido.com
#DEVNET-1882

Which of the following is a valid YANG node type?

In YANG, what is a 'leaf' used for?

What is the primary purpose of YANG in network automation?

Q & A

Appendix

Appendix

GitHub Repo: https://github.com/aaymahaj/DEVNET-1882

NSO Playground: https://developer.cisco.com/codeexchange/github/repo/CiscoDevNet/NSO-Playground-Local-Install/

YANG Documentation: https://developer.cisco.com/docs/nso-guides-6.3/the-yang-data-modeling-language/#yang-introduction

Complete Your Session Evaluations

Complete a minimum of 4 session surveys and the Overall Event Survey to be entered in a drawing to win 1 of 5 full conference passes to Cisco Live 2026.

Earn 100 points per survey completed and compete on the Cisco Live Challenge leaderboard.

Level up and earn exclusive prizes!

Complete your surveys in the Cisco Live mobile app.

Continue your education

Visit the Cisco Showcase for related demos

Book your one-on-one Meet the Engineer meeting

Attend the interactive education with DevNet, Capture the Flag, and Walk-in Labs

Visit the On-Demand Library for more sessions at www.CiscoLive.com/on-demand

Contact me at: aaymahaj@cisco.com

Thank you


CISCO