



The bridge to possible

# Ansible for Collaboration Automation

David Staudt, DevNet Developer Advocate / Principal Engineer

# 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 until February 24, 2023.





# Agenda

- Introduction to Ansible
- Rich UC APIs – Automated & Pipelined!
- CUCM – AXL Requests
- CUCM – AXL SQL Queries
- CUCM – CLI Commands
- Webex – REST / OAuth Tokens w/Vault
- Q&A

# Introduction to Ansible





“Simple, agentless, and powerful open-source IT automation”

# Why Ansible?

- Free, Open Source



- OpenSSH transport



- YAML configuration

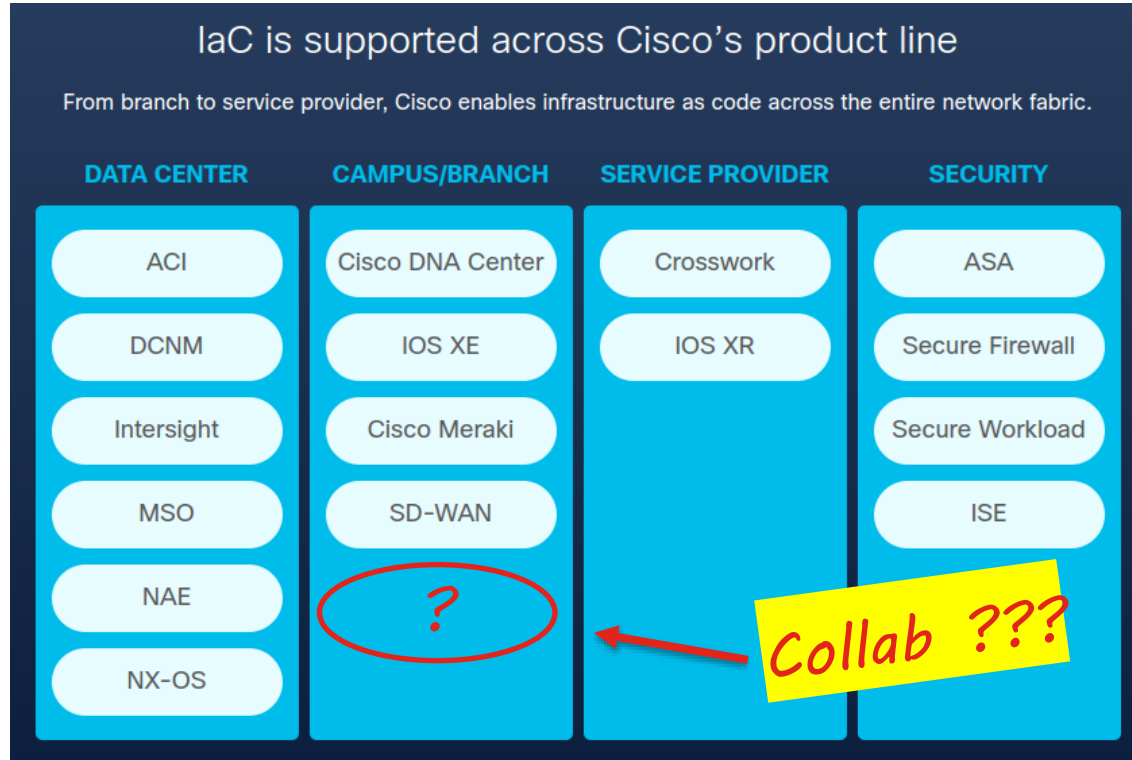


- Based on Python



- Agentless
- Large number of ready to use modules
- Custom modules can be developed if needed
- Run tasks in sequential manner
- Idempotency

# Ansible / Infrastructure as Code at Cisco



<https://developer.cisco.com/iac/>

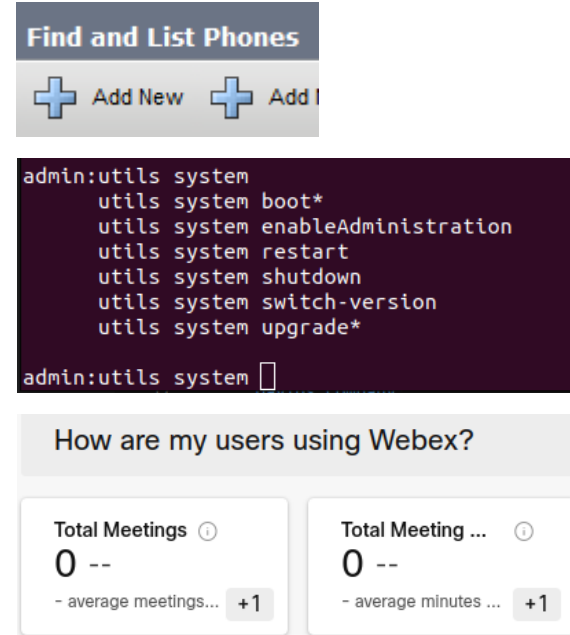
# Ansible CUCM Use-Cases





# Collaboration Use-Cases

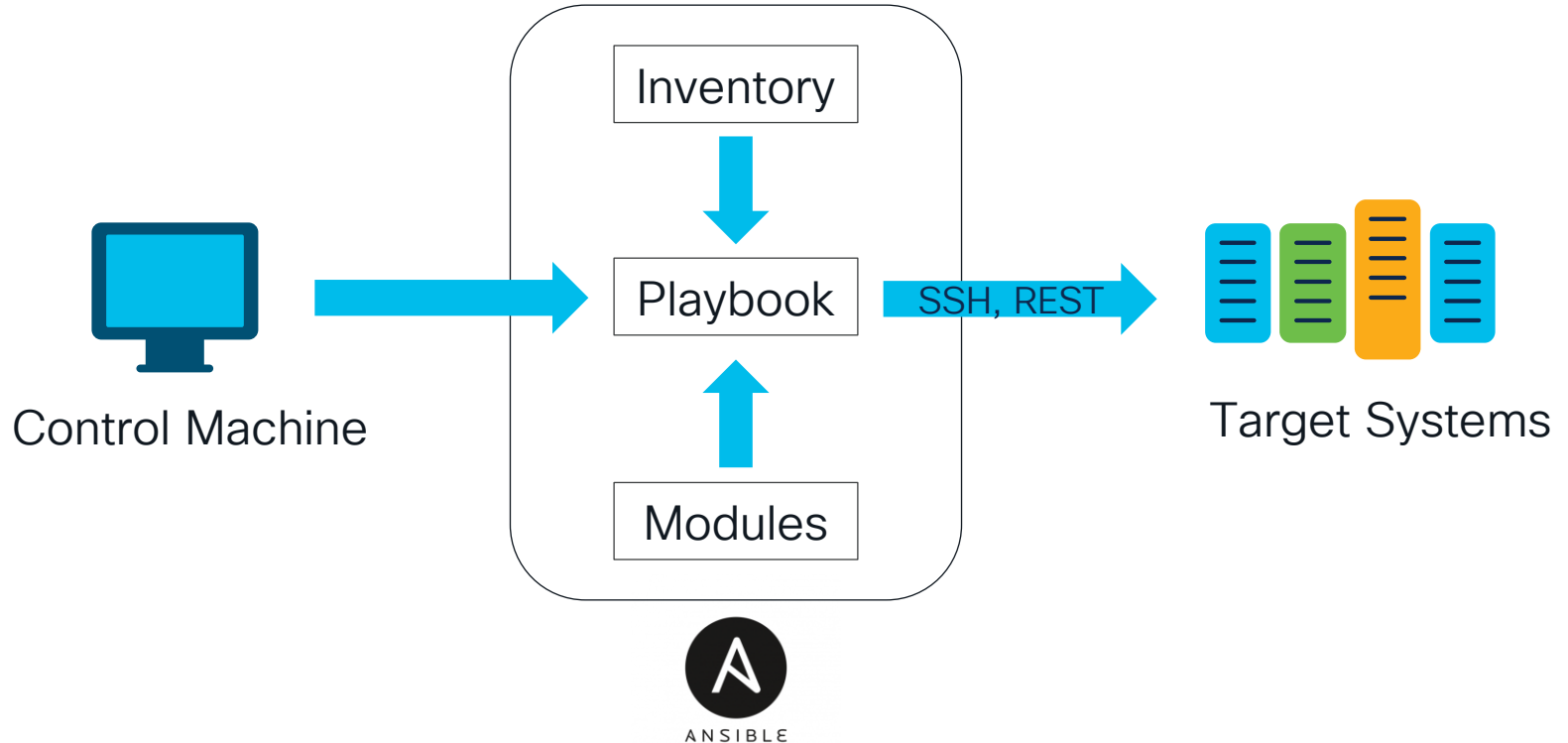
- Provisioning
- Configuration Management
- Security & Compliance
- Orchestration
- Automating daily tasks



# Ansible Basics



# Ansible Architecture



# Ansible Inventory & Playbook

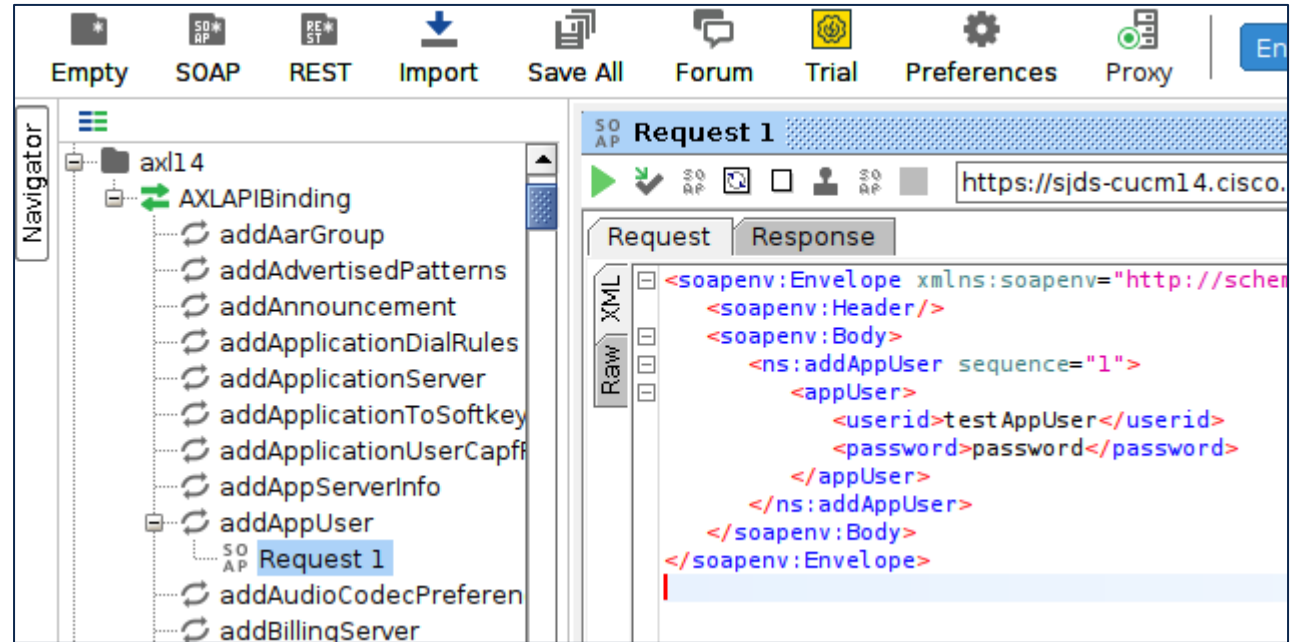
```
! inventory_sandbox.yml U X
1 ---
2 HQCluster:
3   children:
4     publisher:
5       hosts:
6         hq-cucm-pub.abc.inc
7     subscribers:
8       hosts:
9         hq-cucm-sub1.abc.inc
10  vars:
11    axl_user: Administrator
12    axl_password: ciscopsdt
13    cli_user: administrator
14    cli_password: ciscopsdt
15
16 WebexOrgs:
17   hosts:
18     Davids_Company:
19       org_id: Y2lzY29zcGFyazovL3VzL09SR0F0SVpBVElP
20       credential: testauth
21
```

```
! axl_cucm_version_simple.yml X
1 ---
2 - hosts: HQCluster
3   connection: local
4   gather_facts: no
5
6   tasks:
7     - name: Get CUCM version
8       uri:
9         url: "https://{{ ansible_host }}:8443/axl/"
10        method: POST
11        headers:
12          Content-Type: text/xml
13        force_basic_auth: yes
14        user: "{{ hostvars[ansible_host].axl_user }}"
15        password: "{{ hostvars[ansible_host].axl_password }}"
16        body: >
17          <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
18            <soapenv:Body>
19              <ns:getCCMVersion>
20                <processNodeName/>
21              </ns:getCCMVersion>
22            </soapenv:Body>
23          </soapenv:Envelope>
24        validate_certs: false
25        return_content: yes
26        status_code: 200
27        register: response
```

# CUCM AXL



# AXL SQL Toolkit / SoapUI



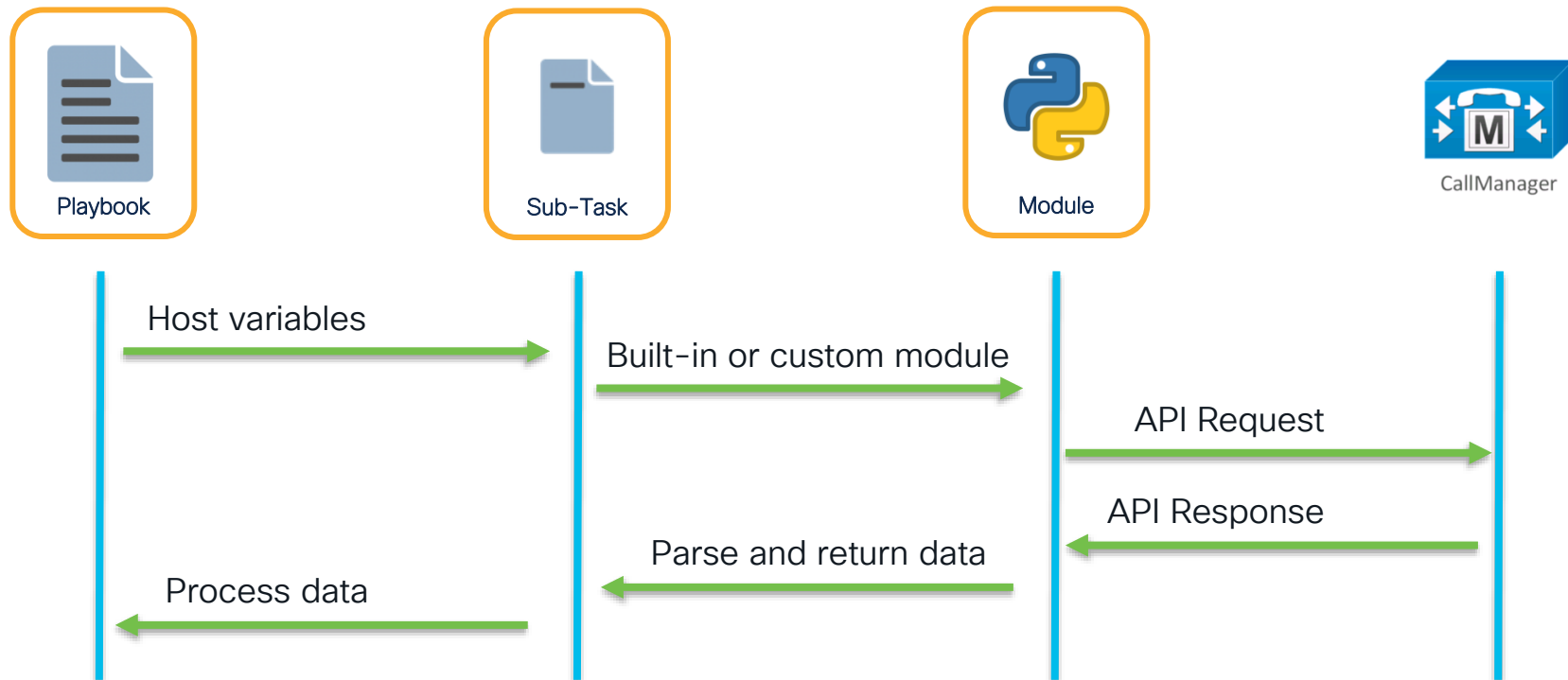
# Demos

- CUCM Version
- Add Line  
(Simple)

GitHub: <https://github.com/CiscoDevNet/axl-ansible-examples>



# Ansible Playbooks / Sub-Tasks / Modules





# Demos

- Add Line
- Provision User  
(Line/Phone/User)

GitHub: <https://github.com/CiscoDevNet/axl-ansible-examples>



# Playbook External Data Access & Looping

```
6 tasks:
7   - name: Read CSV data
8     read_csv:
9       path: USER_DATA.csv
10      register: csv_data
```

USER\_DATA.csv M X

```
1  userid,lastName,password,pin,device_name,model,dn,routePartition
2  testUser1,testUser1,password,123456,SEP000000000001,Cisco 7975,1001,
3  testUser2,testUser2,password,123456,SEP000000000002,Cisco 7975,1002,
4  testUser3,testUser3,password,123456,SEP000000000003,Cisco 7975,1003,
```

```
12   - name: Add bulk users
13     vars:
14       dn: "{{ item.dn }}"
15       routePartition: "{{ item.routePartition }}"
16       device_name: "{{ item.device_name }}"
17       model: "{{ item.model }}"
18       userid: "{{ item.userid }}"
19       lastName: "{{ item.lastName }}"
20       password: "{{ item.password }}"
21       pin: "{{ item.pin }}"
22     include_tasks: axl/addLinePhoneUser.yml
23     loop: "{{ csv_data.list }}"
```

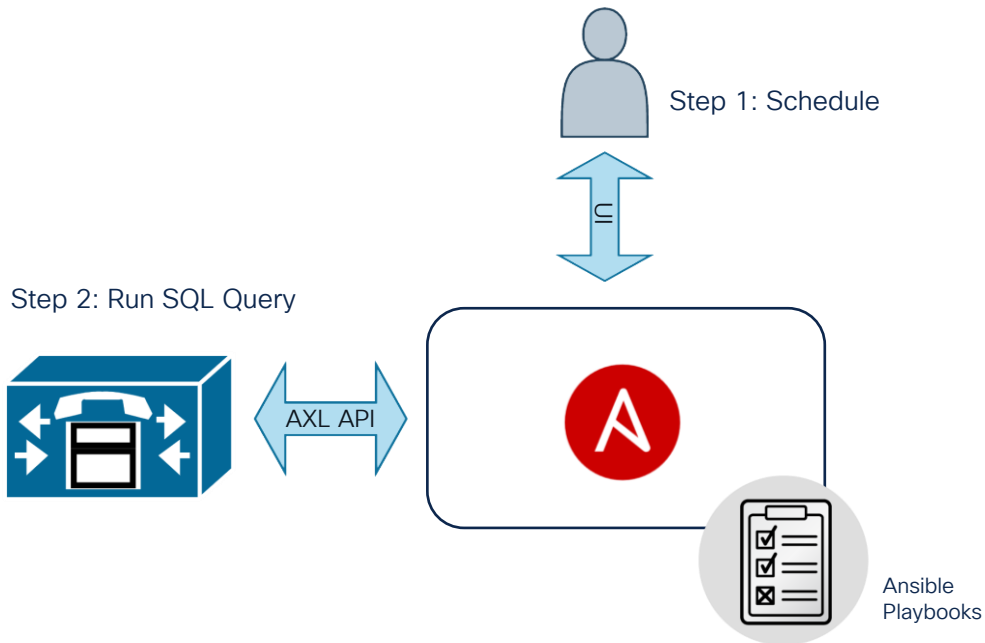
# Demo • Bulk Provisions Users

GitHub: <https://github.com/CiscoDevNet/axl-ansible-examples>



# CUCM SQL Queries – Retrieve Data / Reporting

name	pkid
=====	=====
Administrator	06bae444-79f0-34bc-0b73-042e90ad941b
CCMSysUser	ffd322cd-a1c9-48ce-b23f-6d9475e3547e
WDSysUser	a3d8edb3-8dcd-4e70-a662-dc9afa7f81d3
CCMQRTSysUser	a024f7be-4f36-4373-80dc-a45cb4b891b9
IPMASysUser	d0b9ceb0-d752-46df-96b6-68d37aed70eb
WDSecureSysUser	aaecf22c-ba36-4afd-a8b1-85fb4f02c04f
CCMQRTSecureSysUser	3f2bd34b-c7a1-4b04-a6d4-f75c24c05782
IPMASecureSysUser	bd18e867-2c47-4a60-8740-83c36f178e99
TabSyncSysUser	826888c4-ef7b-48ea-99ba-a86de6c3b369
CUCService	c06dd551-7d3a-4d85-bae9-c450ff03b151
presencevieweradmin	9001d8dc-392b-45b9-97d7-43a663acb50e
KiranAXLUser	b1aa80cb-c9ec-283c-f544-6da59def3f6f
daggett	ce393da4-a17d-8678-6a92-aab64bde837f
testAppUser	0085d951-9f27-a128-e2f7-378783caffc5



# CUCM AXL SQL Queries

The screenshot displays a SOAP client interface with a request and response pane. The request is an `executeSqlQuery` operation, and the response is an `executeSqlQueryResponse` containing a single row of data for the 'Administrator' user.

**Request:**

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header/>
  <soapenv:Body>
    <ns:executeSqlQuery sequence="?">
      <sql>select * from applicationuser</sql>
    </ns:executeSqlQuery>
  </soapenv:Body>
</soapenv:Envelope>
```

**Response:**

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <ns:executeSqlQueryResponse xmlns:ns="http://www.cisco.com/AXL/schema/1.0">
      <return>
        <row>
          <pkid>06bae444-79f0-34bc-0b73-042e90ad941b</pkid>
          <name>Administrator</name>
          <isstandard>t</isstandard>
          <passwordreverse>69c4f936f9cdf45f6bbca2570c3121562</passwordreverse>
          <acloobsubscription>f</acloobsubscription>
          <acloodrefer>f</acloodrefer>
          <aclpresencesubscription>f</aclpresencesubscription>
          <aclunsolicitednotification>f</aclunsolicitednotification>
          <fkmatrix_presence>ad243d17-98b4-4118-8feb-5ff2e1b</fkmatrix_presence>
          <aclallowreplace>f</aclallowreplace>
          <userrank>1</userrank>
        </row>
      </return>
    </ns:executeSqlQueryResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

# Ansible Jinja2 Filters / Custom Filters

```
{{ some_variable | to_json }}  
{{ some_variable | to_yaml }}
```

```
{{ some_variable | to_nice_json }}  
{{ some_variable | to_nice_yaml }}
```

```
{{ some_variable | to_nice_json(indent=2) }}  
{{ some_variable | to_nice_yaml(indent=8) }}
```

```
- name: Extract and save the aXL version string  
  set_fact:  
    axl_version: "{{ cucm_version | regex_search('^\\d+\\.\\d+\\.\\d+\\.\\d+', '\\1') | first }}"
```

- Filters For Formatting Data
- Forcing Variables To Be Defined
- Defaulting Undefined Variables
- Omitting Parameters
- List Filters
- Set Theory Filters
- Random Number Filter
- Shuffle Filter
- Math
- JSON Query Filter
- IP address filter
- Hashing filters
- Combining hashes/dictionaries
- Extracting values from containers
- Comment Filter
- Other Useful Filters
- Combination Filters
- Debugging Filters

# Demo • AXL SQL Query -> CSV

GitHub: <https://github.com/CiscoDevNet/axl-ansible-examples>



# CUCM CLI





# CUCM – Command Line Interface

```
Command Line Interface is starting up, please wait ...

Welcome to the Platform Command Line Interface

VMware Installation:
  4 vCPU: Intel(R) Xeon(R) CPU E5-2640 v3 @ 2.60GHz
  Disk 1: 300GB, Partitions aligned
  16384 Mbytes RAM

admin:show version active
Active Master Version: 14.0.1.10000-20
Active Version Installed Software Options:
cmterm-spectra_IPDECT_14.0v1-sip.k4.cop
cmterm-Ascom_IP_Dect_14.0v1-SIP.k4.cop
cmterm-IPTrade_EK_14.0v1-sip.k3.cop
cmterm-IPTrade_TAD_14.0v1-sip.k3.cop
admin:█
```

- SSH but...quirky
- Username/pass vs SSH keys
- Long(-ish) start-up time
- Not a Unix shell – no Python
- Some commands need interaction
- Textual output (errors?!)
- Not a supported API!

# Custom Ansible Modules

```
module_template.py U X
1  from ansible.module_utils.basic import AnsibleModule
2
3  def run_module():
4      # define available arguments/parameters a user can pass to the module
5      module_args = dict(name=dict(type='str', required=True),
6                          new=dict(type='bool', required=False, default=False))
7      # seed the result dict in the object
8      result = dict(changed=False,
9                   original_message='',
10                   message='')
11      # the AnsibleModule object will be our abstraction working with Ansible
12      module = AnsibleModule(argument_spec=module_args,
13                          supports_check_mode=True)
14      # manipulate or modify the state as needed
15      result['original_message'] = module.params['name']
16      result['message'] = 'goodbye'
17      # AnsibleModule.fail_json() to pass in failure message and the result
18      if module.params['name'] == 'fail me':
19          module.fail_json(msg='You requested this to fail', **result)
20      # in the event of a successful module execution, pass the key/value results
21      module.exit_json(**result)
22
23  def main():
24      run_module()
25
26  if __name__ == '__main__':
27      main()
```

# Calling a Custom Module from a Playbook

! cli\_show\_version\_active.yml X

```
1  ---
2  - hosts: publisher
3    connection: local
4    gather_facts: no
5
6    tasks:
7      - name: Execute CLI command
8        cisco_vos_cli:
9          cli_address: "{{ ansible_host }}"
10         cli_user: "{{ hostvars[ansible_host].cli_user }}"
11         cli_password: "{{ hostvars[ansible_host].cli_password }}"
12         cli_command: show version active
13         register: response
14
15      - debug:
16        msg: |
17          Raw: {{ response.output['raw'] }}
18          Tokenized: {{ response.output['tokenized'] | to_nice_json }}
19          Active version: {{ response.output['tokenized'][0][3] }}
```

# Demo

- CLI show active version

GitHub: <https://github.com/CiscoDevNet/axl-ansible-examples>



# CLI User Interaction - Paramiko Expect

```
Command Line Interface is starting up, please wait ...
```

```
Welcome to the Platform Command Line Interface
```

```
VMware Installation:
```

```
4 vCPU: Intel(R) Xeon(R) CPU E5-2640 v3 @ 2.60GHz
```


```
Disk 1: 300GB, Partitions aligned
```

```
16384 Mbytes RAM
```

```
admin:set cli session timeout 30
```

```
Continuing with this operation will change the session timeout  
for all new CLI sessions to 30 minutes.
```

```
Continue (y/n)? ☐
```



```
cli_responses:  
- expect: "Continue (y/n)?"  
  response: y  
  timeout: 10
```

# Webex REST



# Webex REST is Easy – OAuth Tokens are Harder...

```
uri:
  url: "https://webexapis.com/v1/licenses?{{ org_id }}"
  method: GET
  headers:
    Authorization: "Bearer {{ webex_token }}"
  return_content: yes
  status_code:
    - 200
register: response
```

- Webex tokens: Bot or OAuth2 User
- Bots – No Admin Operations!
- Webex Integrations Allow Apps to Authenticate
- OAuth2 – Requires ‘manual’ User Login w/Browser
- Ansible: CLI / Scheduled Tasks
- Tokens Expire (~14 days)

# Webex OAuth Token Refresh

```
{  
    "access_token": "ZDI3MGEyYzQtNmFlNS00NDNhLWF7NjEwMTQzMDEyOTUxNTg6MDAwMA==",  
    "expires_in": 1209600, //seconds  
    "refresh_token": "MDEyMzQ1NiJc40TAxmjm0NTY3ODkwMmVlZW8tLTk1LWVkbG9jaXo= ",  
    "refresh_token_expires_in": 7776000 //seconds  
}
```



Requires a long-running process / server

```
expires_in: 14 days
refresh_token_expires_in: 90 days
```



# HashiCorp Vault + oauthapp Plugin



HashiCorp  
**Vault**

*“Manage Secrets &  
Protect Sensitive Data”*



**vault-plugin-secrets-oauthapp**

# Using Vault Secrets with Ansible

```
vars:
  secret: "secret=oauthapp/creds/{{ credential }} token={{ lookup('hashi_vault', secret) }}"
  credentials: "{{ lookup('hashi_vault', secret) }}"
  webex_token: "{{ credentials | json_query('access_token') }}"
uri:
  url: "https://webexapis.com/v1/licenses?{{ org_id }}"
  method: GET
  headers:
    Authorization: "Bearer {{ webex_token }}"
```

- Webex admin logs in once
- Token is refreshed automatically – forever\*
- SysAdmins never see the Webex admin password
- SysAdmins can revoke Vault token – Webex admin unaffected
- Use Vault for all your secrets!

# Demo • Webex REST + Vault

GitHub: <https://github.com/CiscoDevNet/axl-ansible-examples>



# Q&A

# Resources

- [developer.webex.com](https://developer.webex.com)
- [cisco.devnet/axl-ansible-examples](https://cisco.devnet/axl-ansible-examples)
- [puppetlabs/vault-plugin-secrets-oauthapp](https://puppetlabs/vault-plugin-secrets-oauthapp)
- [Cisco with Ansible](#)
- [Ansible Code on DevNet Code Exchange](#)
- [Ansible on DevNet Learning Labs](#)

# Complete your Session Survey

- Please complete your session survey after each session. Your feedback is important.
- All surveys can be taken in the Cisco Events Mobile App or by logging in to the Session Catalog and clicking the "Attendee Dashboard" at <https://www.ciscolive.com/emea/learn/sessions/session-catalog.html>



# Continue Your Education



Visit the Cisco Showcase for related demos.



Book your one-on-one Meet the Engineer meeting.



Attend any of the related sessions at the DevNet, Capture the Flag, and Walk-in Labs zones.



Visit the On-Demand Library for more sessions at [ciscolive.com/on-demand](https://ciscolive.com/on-demand).



The bridge to possible

# Thank you

CISCO *Live!*



CISCO *Live!*

ALL IN