The bridge to possible

# Websockets and webhooks: Embed network intelligence into your applications

Adrian ILIESIU, Senior Technical Leader
@aidevnet

cisco Live!

DEVNET-2262

# Cisco Webex App

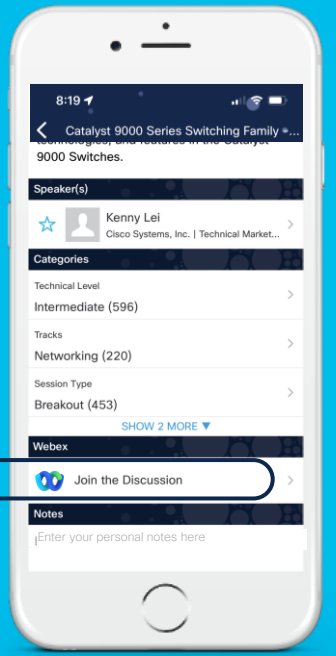
## Questions?

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

- WebSockets
- Webhooks
- Conclusion
- Resources

# WebSockets

# What is a WebSocket?

- Communication protocol used to send/receive data on the Internet (RFC6455)
- Like HTTP but much better and more efficient (TCP ports 80 & 443)
- Persistent 2-way connection between client <--> server
- Easy to build real-time applications:
  - Online games
  - Financial trading
  - Collaboration apps
  - Notifications
  - Chat

## HTTP

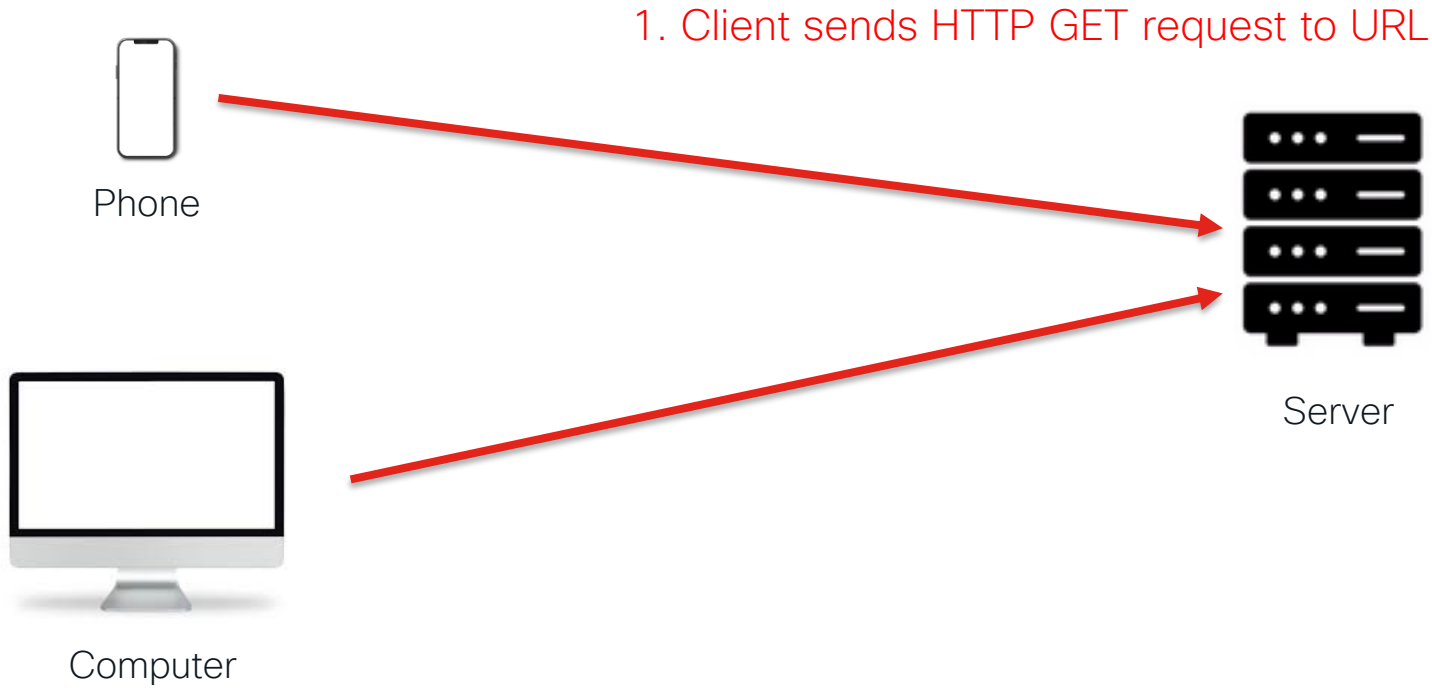
- Half-duplex (like walkie-talkie)
- Traffic flows in 1 direction at a time
- Connection closes after 1 request/response
- Large headers (1000s of bytes)
- 150ms to establish new TCP connection for each HTTP message
- Polling overhead

1. Request from client to server
2. Response from server to client

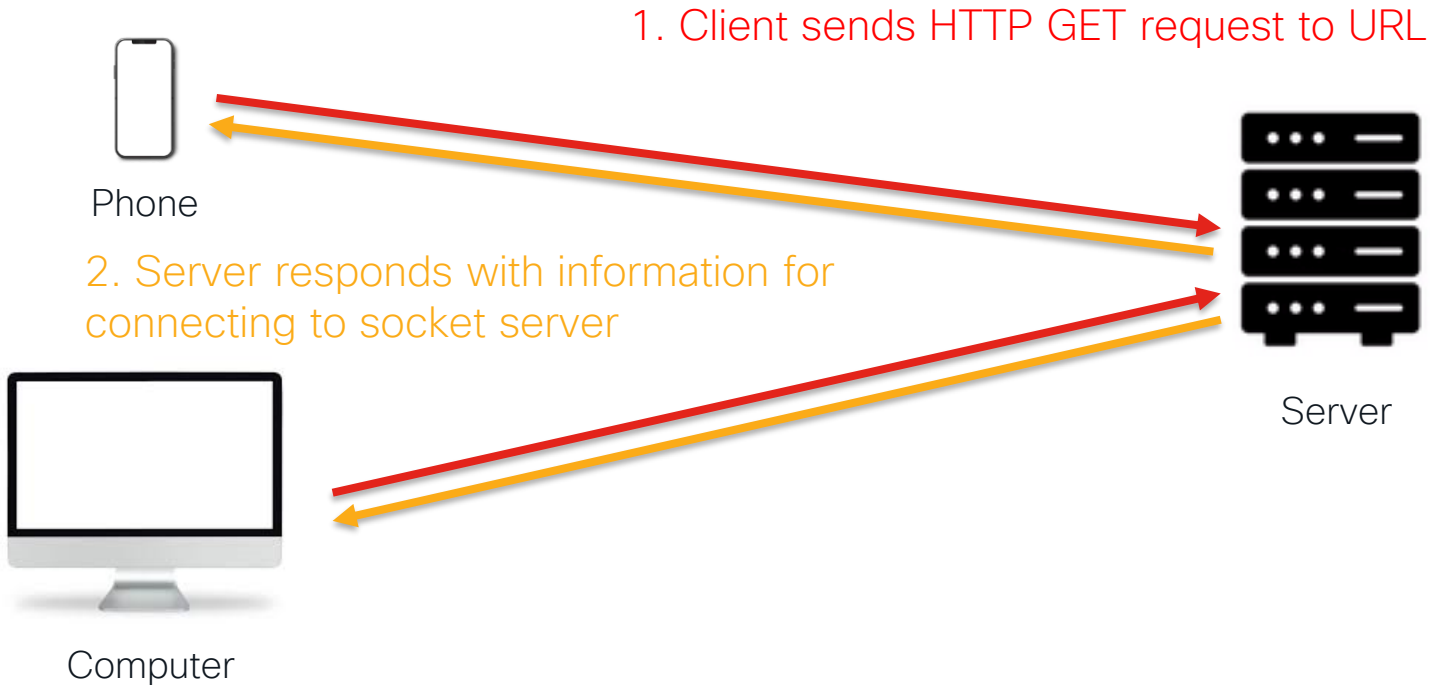
## WebSocket

- Full-duplex (like phone)
- Bi-directional traffic flow
- Connection stays open
- Uses “frames” (2 bytes)
- 50ms for message transmission
- No polling overhead
- Both client and server are simultaneously “transmitting” and “listening”

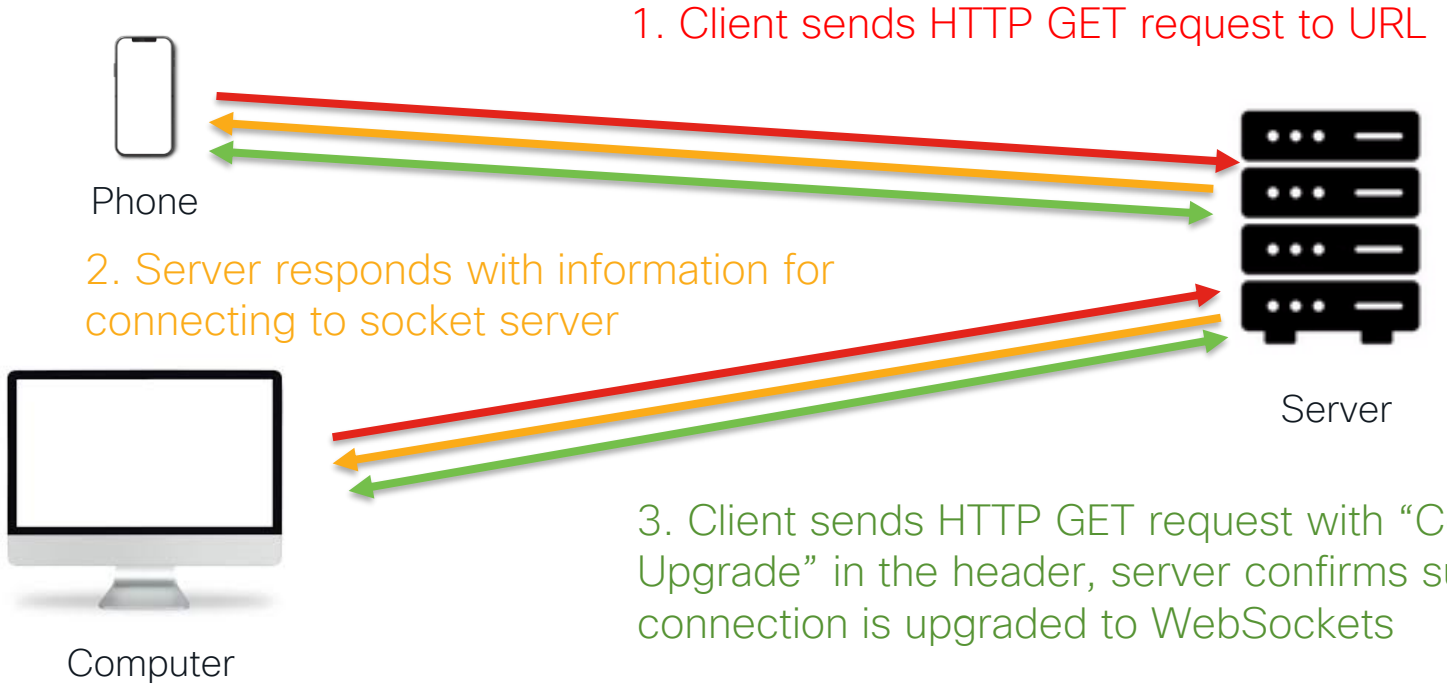
# How do WebSockets work?



# How do WebSockets work?



# How do WebSockets work?



# The handshake

- Client sends:

--- request header ---

GET /socketW2OM/Potf4WGMUt/vFMgQ7w6EUOCBonqeYK2jFvCPDKF5LFcu5ea6v/WcccAuRbm+qKhjvEvwu/ukymvGuI96GOHZ1fFr6Sce041cuChZCfz0JXog5y9iUKIofi84UswURGylKaw2fhedCJfj461cX/BhsHypvqM1AivQA1gwRg= HTTP/1.1

Upgrade: websocket

Host: 10.10.10.177

Origin: https://10.10.10.177

Sec-WebSocket-Key: FtK1+TbzPV4dHiuPQW+Z/Q==

Sec-WebSocket-Version: 13

Connection: Upgrade

- Server responds:

--- response header ---

HTTP/1.1 101 Switching Protocols

Server: nginx/1.7.10

Date: Sat, 14 Jan 2023 05:55:11 GMT

Connection: upgrade

Sec-WebSocket-Accept: FzXeL1fziR8iOHfBd2szRy7SKZ0=

Upgrade: websocket

Strict-Transport-Security: max-age=31536000; includeSubDomains

X-Frame-Options: SAMEORIGIN

Content-Security-Policy: block-all-mixed-content; base-uri 'self'; default-src 'self'; script-src 'self' 'nonce-xxUytQEYl3o8U8bZX6PRnfwdh1c139fQ'; style-src 'self' 'nonce-xxUytQEYl3o8U8bZX6PRnfwdh1c139fQ'; img-src 'self'; connect-src 'self'; font-src 'self'; object-src 'self'; media-src 'self'; form-action 'self'; frame-ancestors 'self';

# Advantages and disadvantages

- Advantages:
  - Great for real-time applications
  - Low latency
  - Low overhead
  - Less traffic
- Disadvantages:
  - Not good for retrieving old historical data
  - Not largely implemented for network monitoring
  - Varied support in Internet browsers

# Cisco Open NX-OS – Login function

```
nxos_login:
        address: "10.10.10.177"
        username: "admin"
        password: "admin"
```

```
def nxosLogin():

        response = requests.post(
                'https://' + config['nxos_login']['address'] + '/api/aaaLogin.json',
                headers = {'Content-Type': 'application/json'},
                data = json.dumps(nxosLoginTemplate),
                verify = False
        )

        responseDict = json.loads(response.text)
        token = responseDict['imdata'][0]['aaaLogin']['attributes']['token']

        message = "NXOS Login successful."
        writeLog(message)

        return token
```

# Cisco Open NX-OS – Login function (cont.)

```
def nxosLogin():

        response = requests.post(
                'https://' + config['nxos_login']['address'] + '/api/aaaLogin.json',
                headers = {'Content-Type': 'application/json'},
                data = json.dumps(nxosLoginTemplate),
                verify = False
        )

        responseDict = json.loads(response.text)
        token = responseDict['imdata'][0]['aaaLogin']['attributes']['token']

        message = "NXOS Login successful."
        writeLog(message)

        return token
```

```
{
        "aaaUser" : {
                "attributes" : {
                        "name" : "admin",
                        "pwd" : "admin"
                }
        }
}
```

# Cisco Open NX-OS – Login function (cont.)

```
def nxosLogin():

        response = requests.post(
                'https://' + config['nxos_login']['address'] + '/api/aaaLogin.json',
                headers = {'Content-Type': 'application/json'},
                data = json.dumps(nxosLoginTemplate),
                verify = False
        )

        responseDict = json.loads(response.text)
        token = responseDict['imdata'][0]['aaaLogin']['attributes']['token']

        message = "NXOS Login successful."
        writeLog(message)

        return token
```

# Cisco Open NX-OS – Subscribe function

```
monitored_objects:
  - /api/mo/sys/intf/phys-[eth1/3]
  - /api/mo/sys/intf/phys-[eth1/11]
  - /api/node/mo/sys/bd/bd-[vlan-101]
```

```
def subscribe(loginToken):
    subIds = []

    for sub in config['monitored_objects']:
            response = requests.get(
                "https://" + config['nxos_login']['address'] + sub + ".json?subscription=yes",
                headers = {'Cookie': "APIC-cookie=" + loginToken},
                verify = False
            )
            subIds.append(json.loads(response.text)['subscriptionId'])

    message = "Subscription successful. Subscription IDs:\n"
    for subid in subIds:
            message = message + subid + "\n"
    writeLog(message)
```

# Cisco Open NX-OS – Subscribe function (cont.)

```
def subscribe(loginToken):
    subIds = []

    for sub in config['monitored_objects']:
            response = requests.get(
                "https://" + config['nxos_login']['address'] + sub + ".json?subscription=yes",
                headers = {'Cookie': "APIC-cookie=" + loginToken},
                verify = False
            )
            subIds.append(json.loads(response.text)['subscriptionId'])

    message = "Subscription successful. Subscription IDs:\n"
    for subid in subIds:
            message = message + subid + "\n"
    writeLog(message)
```

# Cisco Open NX-OS – Refresh function

```
def refresh():
    while True:
            time.sleep(55)

            with open(pathSubscriptionIds, "r") as handle:
                subscriptionIds = json.load(handle)

            loginToken = nxosLogin()

            message = "Subscription Refresh successful. Refreshed subscription IDs: \n"
            for sub in subscriptionIds:
                response = requests.get(
                        "https://" + config['nxos_login']['address'] + "/api/subscriptionRefresh.json?id=" + sub,
                        headers = {'Cookie': "APIC-cookie=" + loginToken},
                        verify = False
                )
                print(response.text)
                if not json.loads(response.text)['imdata']:
                        message = message + sub + "; "
                else:
                        message = message + "Subscription " + sub + " could not be refreshed.\n"

            writeLog(message)
```

# Cisco Open NX-OS – Main function

```
import websocket
import threading
import ssl
import os

if __name__ == "__main__":
        loginToken = nxosLogin()

        refreshThread = threading.Thread(target=refresh)
        refreshThread.start()

        websocket.enableTrace(True)
        ws = websocket.WebSocketApp("wss://" + config['nxos_login']['address'] + "/socket" + loginToken,
                                    on_message = on_message,
                                    on_error = on_error,
                                    on_close = on_close,
                                    on_open = on_open)

        ws.run_forever(sslopt={"cert_reqs": ssl.CERT_NONE})
```

```
def on_message(ws, message):
        outputToFile(message)
```

```
def on_error(ws, error):
        writeLog(error)
```

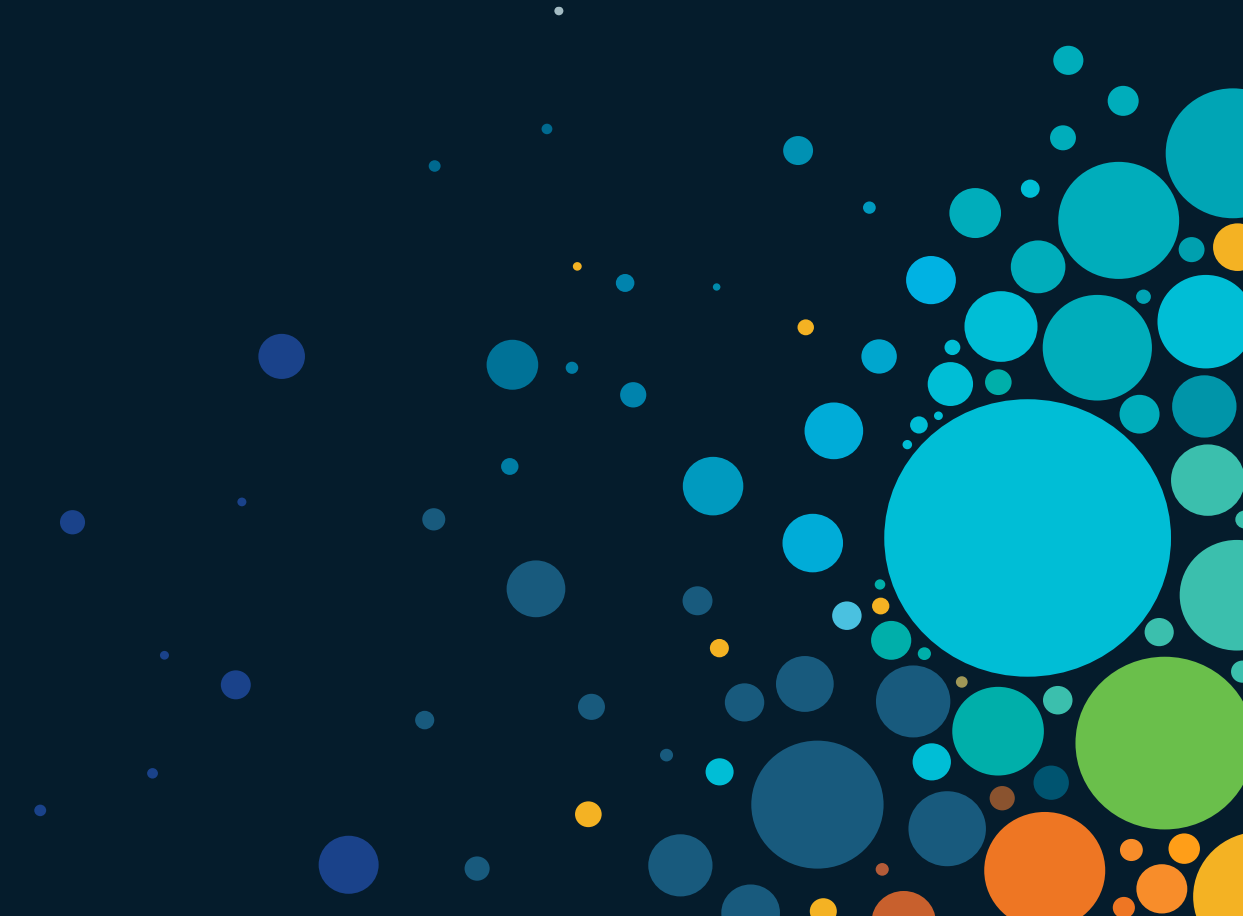
```
def on_open(ws):
        subscribe(loginToken)
```

```
def on_close(ws):
        os.remove(pathSubscriptionIds)
        message = "Socket was closed."
        writeLog(message)
```

# Cisco Open NX-OS – Subscription output


```
{
        "subscriptionId": ["18374686505424650242"],
        "imdata": [{
                "l1PhysIf": {
                        "attributes": {
                                "adminSt": "up",
                                "childAction": "",
                                "dn": "sys/intf/phys-[eth1/11]",
                                "modTs": "2023-01-22T06:04:36.208+00:00",
                                "rn": "",
                                "status": "modified"
                        }
                }
        }]
}
{
        "subscriptionId": ["18374686505424650243"],
        "imdata": [{
                "l2BD": {
                        "attributes": {
                                "childAction": "",
                                "dn": "sys/bd/bd-[vlan-101]",
                                "operSt": "up",
                                "rn": "",
                                "status": "modified"
                        }
                }
        }]
}
```

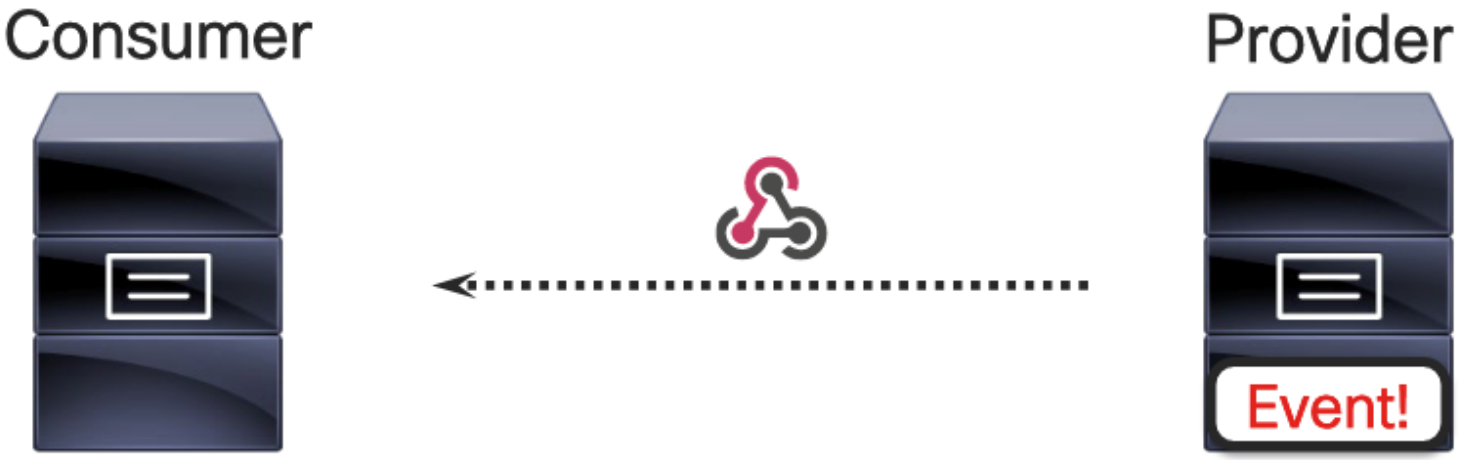
# Webhooks



# What is a Webhook?

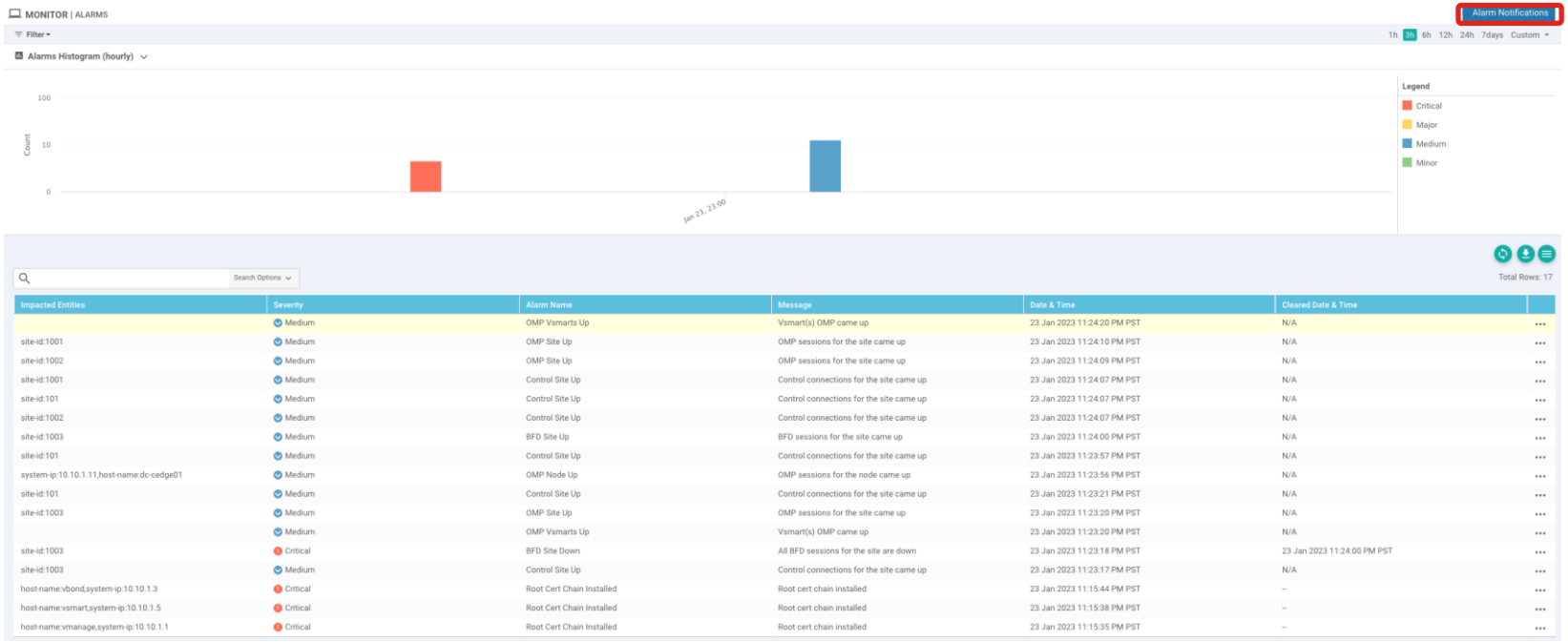
- Webhooks are automated messages sent from apps when something happens
- Message is sent to a preconfigured URL when an event get triggered
- Faster and less resource intensive than polling
- Like SMS notifications
- Used for:
  - Sending small amounts of data
  - Trigger automation workflows

# How do Webhooks work?



# Webhook configuration for Cisco SD-WAN

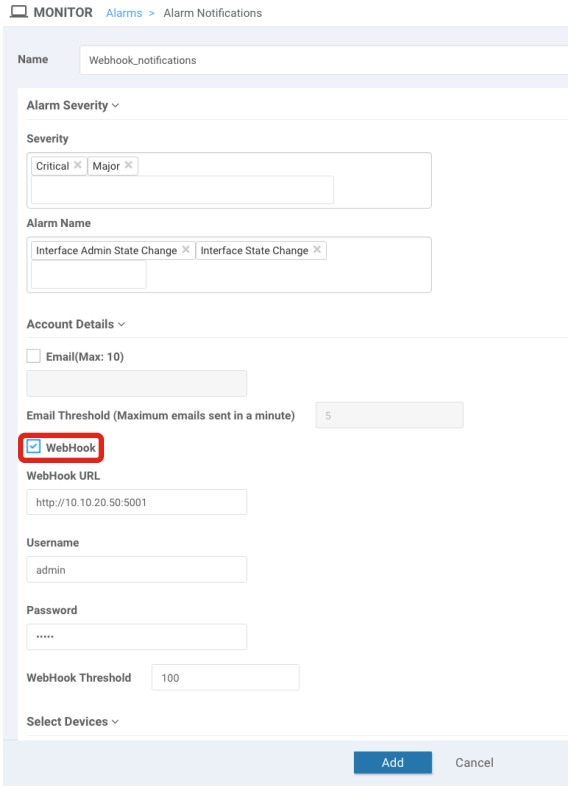
- Step 1: Select “Alarm Notifications” from “Monitor -> Alarms”



# Webhook configuration for Cisco SD-WAN (cont.)

Step 2:

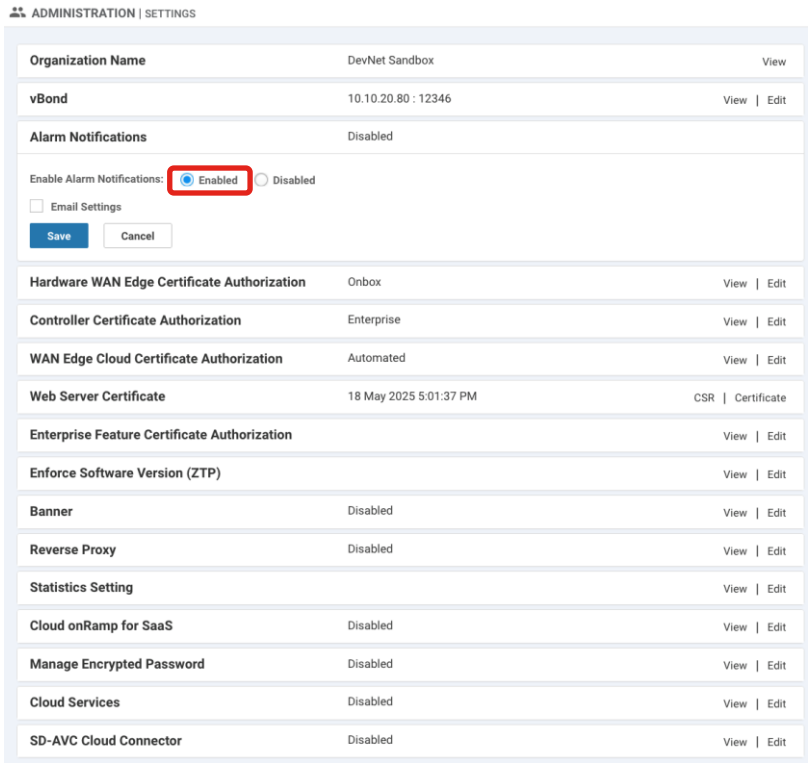
- Enter a name for the webhook
- Select severity level (all, critical, major, medium, minor)
- Select alarm name (disk usage, process restart, etc.)
- Enable WebHook checkbox
- Provide the WebHook URL, username and password
- Select devices (all, custom)



# Webhook configuration for Cisco SD-WAN (cont.)

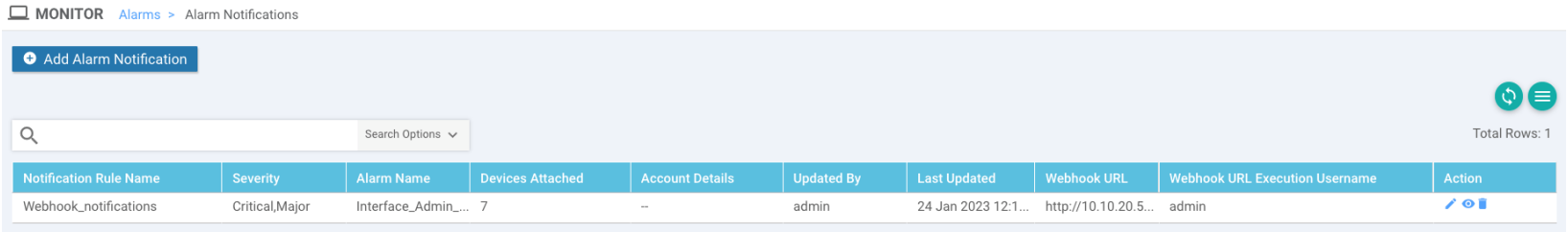
Step 3:

- Enable “Alarm Notifications” under the Administration Settings

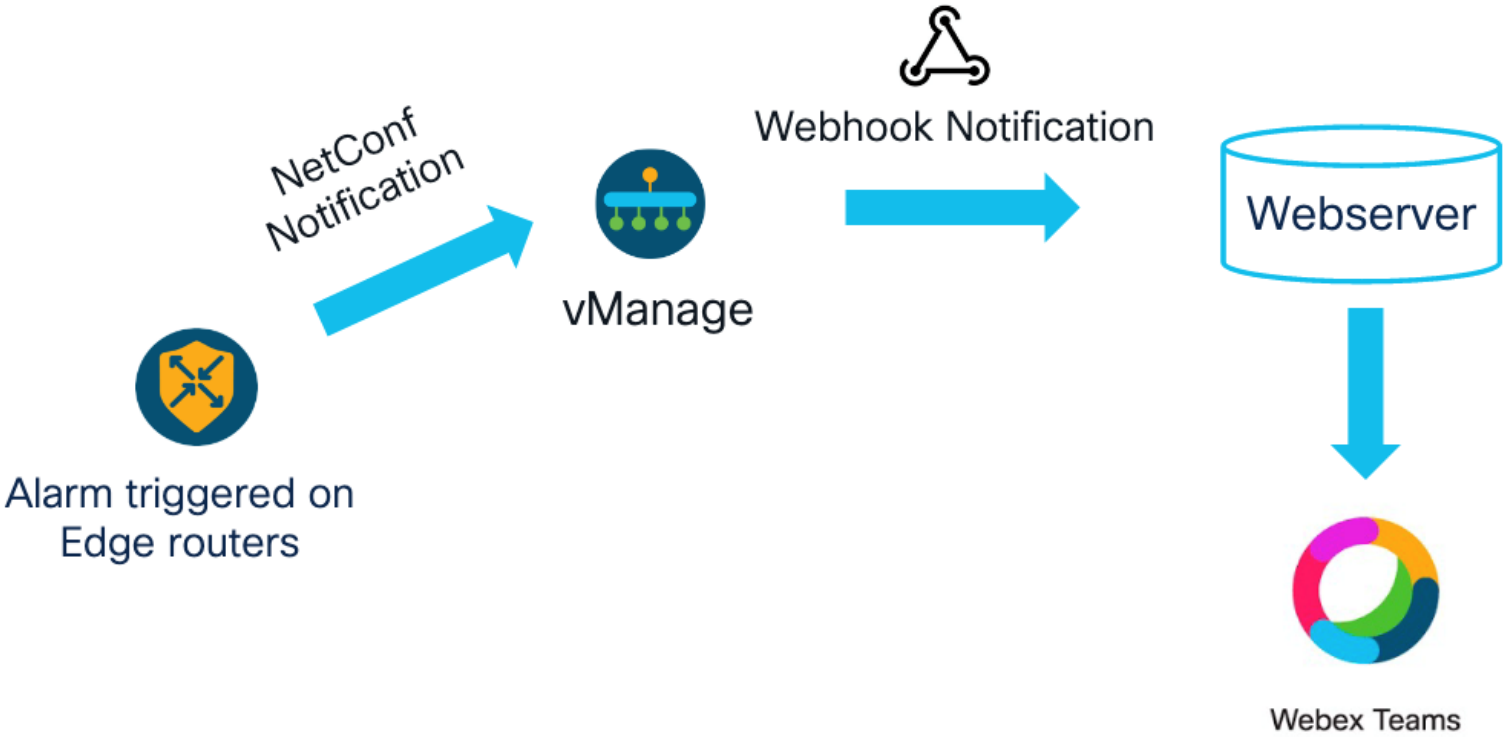


# Webhook configuration for Cisco SD-WAN (cont.)

- Step 4: Verify list of webhooks under “Alarms -> Alarm Notifications”



# Webhook notifications flow



# Sample alert


```
{
    "entry_time": 1557638912000,
    "severity": "Medium",
    "rule_name_display": "Interface_State_Change",
    "severity_number": 3,
    "component": "VPN",
    "values_short_display": [
        {
            "if-name": "GigabitEthernet4",
            "host-name": "BR2-CSR1000v",
            "system-ip": "1.1.1.6",
            "new-state": "up"
        }
    ],
    "devices": [
        {
            "system-ip": "1.1.1.6"
        }
    ],
    "eventname": "interface-state-change",
    "receive_time": 1557638912888,
    "statcycletime": 1557638912000,
    "values": [
...
```

```
...
    "values": [
        {
            "if-name": "GigabitEthernet4",
            "vpn-id": "10",
            "host-name": "BR2-CSR1000v",
            "system-ip": "1.1.1.6",
            "new-state": "up"
        }
    ],
    "cleared_events": [
        "8459e3a0-5bea-4370-ab57-6f45f8022d66"
    ],
    "rulename": "interface-state-change",
    "active": false,
    "message": "The interface oper-state changed to up",
    "type": "interface-state-change",
    "acknowledged": false,
    "uuid": "7a514a95-7c24-4348-b7e9-8d6775a3bc36"
}
```

# Setup webhook receiver server

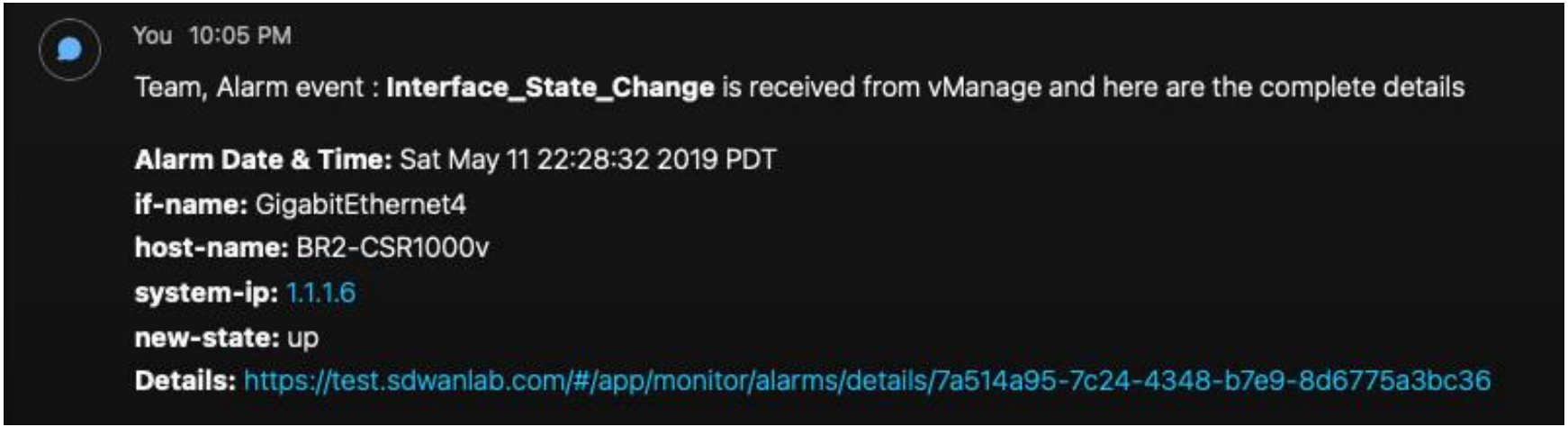
```
from flask import Flask, request, jsonify
from webexteamssdk import WebexTeamsAPI
import json
import os
import datetime
import pytz

bearer_token = os.environ.get("bearer_token")
room_id = os.environ.get("room_id")

app = Flask(__name__)

@app.route('/',methods=['POST'])
def alarms():
        try:
                PDT = pytz.timezone('America/Los_Angeles')
                data = json.loads(request.data)

                message = '''Team, Alarm event : **''' + data['rule_name_display'] + '''** is received from
vManage and here are the complete details <br><br>'''
```


```
{
        "entry_time": 1557638912000,
        "severity": "Medium",
        "rule_name_display": "Interface_State_Change",
        "severity_number": 3,
        "component": "VPN",
        "values_short_display": [
                {
                        "if-name": "GigabitEthernet4",
                        "host-name": "BR2-CSR1000v",
                        "system-ip": "1.1.1.6",
                        "new-state": "up"
                }
        ],
        "devices": [
                {
                        "system-ip": "1.1.1.6"
                }
        ],
...
}
```

# Setup webhook receiver server (cont.)

```
                temp_time = datetime.datetime.utcfromtimestamp(data['entry_time']/1000.)
                temp_time = pytz.UTC.localize(temp_time)
                message = message + '**Alarm Date & Time:** ' +
                temp_time.astimezone(PDT).strftime('%c') + ' PDT'
                temp = data['values_short_display']
                for item in temp:
                        for key, value in item.items():
                                message = message + '<br> **' + key + ':** ' + value

                message = message + '<br> **' + 'Details:' + '** ' +
        "https://test.sdwanlab.com/#/app/monitor/alarms/details/" + data['uuid']

                api = WebexTeamsAPI(access_token=bearer_token)
                res=api.messages.create(roomId=room_id, markdown=message)

        except Exception as exc:
                print(exc)
                return jsonify(str(exc)), 500
        return jsonify("Message sent to Webex Teams"), 200

if __name__ == '__main__':
        app.run(host='0.0.0.0', port=5001, debug=True)
```

# Webex Teams message



# 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.

# Conclusions

- WebSockets and Webhooks are popular web technologies
- Websockets supported by: Cisco Open NX-OS, Cisco ACI, Cisco ISE, Webex Teams
- Webhooks supported by: Cisco SD-WAN, Meraki, Cisco DNA Center, ThousandEyes, Webex Teams, Cisco Intersight, Cisco SecureX
- Easy to integrate JSON formatted status and alert information received over websockets and webhooks
- Embed this information into applications to make them aware of infrastructure status in real-time

# Resources

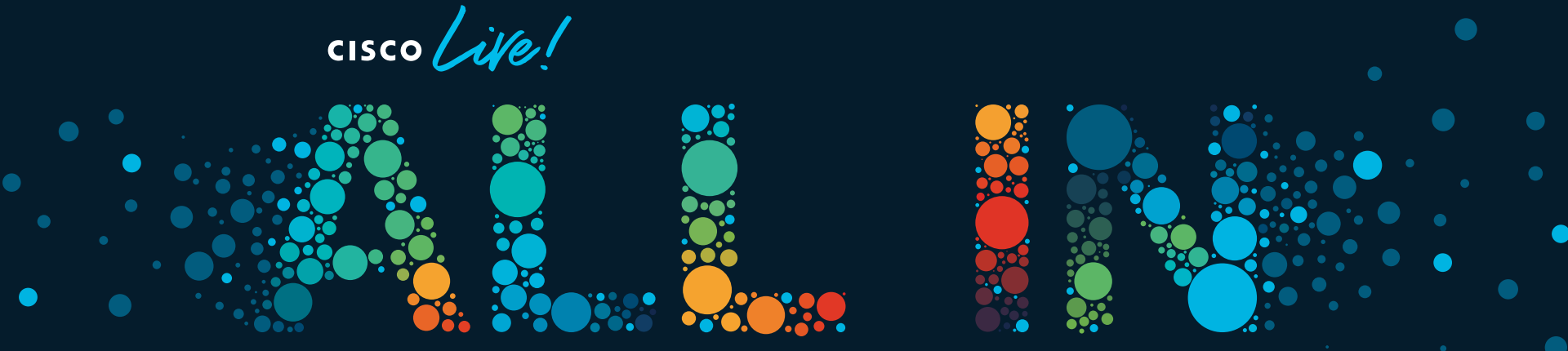
- https://developer.cisco.com/codeexchange
- https://developer.cisco.com/docs/nx-os/#!cisco-nx-api-websocket-notifications
- https://developer.cisco.com/learning/labs/sd-wan-webhooks/getting-started-with-webhooks-on-cisco-sd-wan/

# Q&A


The bridge to possible

# Thank you

cisco Live!


CISCO Live!
ALL IN