cisco

Cisco live!

January 29 - February 2, 2018 · Barcelona

DEVNET-1480

Empower your testing with Cisco Test Automation Solution

Featuring pyATS & Genie

Siming Yuan, Technical Leader, Engineering, Cisco

Jean-Benoit Aubin, Engineer, Software Engineering, Cisco

Sedy Yadollahi, Manager, Software Engineering, Cisco

Ramesh Yeevani-Srinivas, Director, Engineering, Cisco

Cisco live!

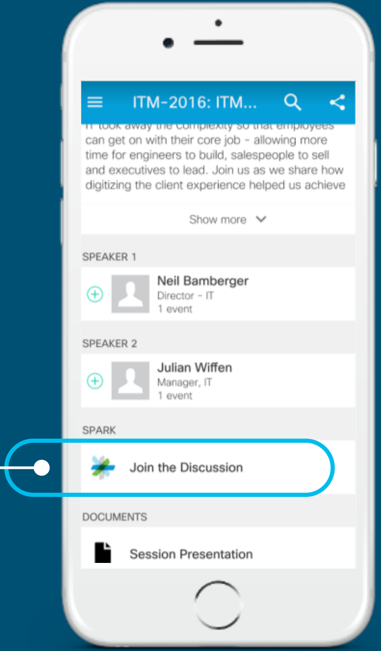
Cisco Spark

Questions?

Use Cisco Spark to communicate with the speaker after the session

How

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


ITM-2016: ITM...

IT took away the complexity so that employees can get on with their core job - allowing more time for engineers to build, salespeople to sell and executives to lead. Join us as we share how digitizing the client experience helped us achieve

Show more

SPEAKER 1

Neil Bamberger
Director - IT
1 event

SPEAKER 2

Julian Wiffen
Manager, IT
1 event

SPARK

Join the Discussion

DOCUMENTS

Session Presentation

cs.co/ciscolivebot#DEVNET-1480

Agenda

- Introduction, Background
- Solution Overview
- pyATS Features At-a-Glance
- Genie Library, SDK
- Installation, Getting Started
- Examples & Resources
- Roadmap, Upcoming Releases

Intuitive Test Automation For Intuitive Networks

pyATS, 2014 - present

- Launched internally in Cisco engineering late 2014
- Quickly became the most adopted test framework within Cisco
- Running in sanity, regression, solution labs, etc.

3000+ developers

5,000,000+ LoC

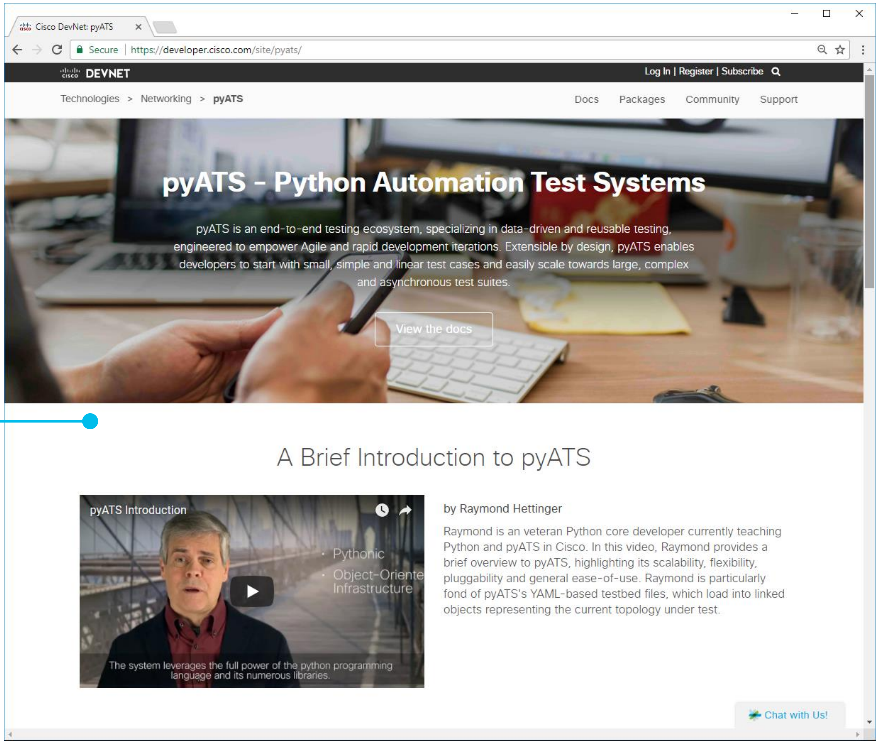
$\frac{\text{1,000,000+ runs}}{\text{month}}$

“Can we share this with our customers?”

- Dave Wapstra, Architect, Solution Integration, Cisco

DevNet: pyATS

developer.cisco.com/site/pyats/


Cisco DevNet: pyATS
Secure | https://developer.cisco.com/site/pyats/
DEVNET
Log In | Register | Subscribe
Technologies > Networking > pyATS
Docs
Packages
Community
Support
View the docs
pyATS Introduction
Chat with Us!

pyATS – Python Automation Test Systems

pyATS is an end-to-end testing ecosystem, specializing in data-driven and reusable testing, engineered to empower Agile and rapid development iterations. Extensible by design, pyATS enables developers to start with small, simple and linear test cases and easily scale towards large, complex and asynchronous test suites.

A Brief Introduction to pyATS

- Pythonic
- Object-Oriented Infrastructure

The system leverages the full power of the python programming language and its numerous libraries.

by Raymond Hettinger

Raymond is an veteran Python core developer currently teaching Python and pyATS in Cisco. In this video, Raymond provides a brief overview to pyATS, highlighting its scalability, flexibility, pluggability and general ease-of-use. Raymond is particularly fond of pyATS's YAML-based testbed files, which load into linked objects representing the current topology under test.

Why?

Traditional Test Automation

- *linear* and *monolithic*
- functional based
- single-purpose
- favors functional, test automation specialist teams

New Requirements

- *heterogeneous* and *polymorphic*
- dynamic and data-driven
- pluggable and extendable
- cross-functional teams with SMEs

LEGO

CATERHAM
CATERHAM

pyATS Features At-a-Glance

Agnostic: Multi-Vendor Support

Python 3.4 +

Extensible Framework

Independent Modules

Tcl Library Reuse

Configurable Report

Asynchronous Execution

YAML-defined, Object-based Topology

Device Connection Management

Test cases, Sections, Steps

Selective Execution

Data-Driven Test Cases

Reusable Tests, Looping

Parametrized Test cases

Pause & PDB on [Anything]

Pre/Post/Exception Processors

Device Bring-up/Clean Management

Test Script Example

```yaml
devices:
  ios-1:
    type: ios
    connections:
      console:
        protocol: telnet
        ip: 1.1.1.1
        port: 2003

  ios-2:
    #...

topology:
  ios1:
    interfaces:
      Ethernet0/0:
        ipv4: 10.10.10.1/24
        link: link-1

  ios2:
    interfaces:
      Ethernet0/0:
        ipv4: 10.10.10.2/24
        link: link-1
```

```python
import re
from ats import aetest

class CommonSetup(aetest.CommonSetup):
    @aetest.subsection
    def connect_to_devices(self, testbed):
        for device in testbed: testbed[device].connect()

@aetest.loop(device = ('ios-1', 'ios-2'))
class TestPing(aetest.Testcase):

    @aetest.test.loop(destination = ('10.10.10.1', '10.10.10.2'))
    def ping(self, device, testbed, destination):
        try:
            result = testbed[device].ping(destination)
        except Exception as e:
            self.failed('Ping {} from device {} failed with error: '
                        '{}'.format(destination, device), from_exception = e)

        match = re.search(r'Success rate is (?P<rate>\d+) percent', result)
        assert int(match.group('rate')) == 100, \
            'Ping {} with success rate of {}%'.format(destination,
                                                     success_rate)
```

Genie

Provides ***feature-centric*** object models

- Focuses development effort on writing test cases & suites
- Shields the end scripter from explicit CLI/YANG-RPCs

Objects are ***agnostic***

- Works across management interfaces: CLI, YANG, XML, etc.
- Handles feature differences between images, releases, platforms, etc.

Genie is ***plug & play***

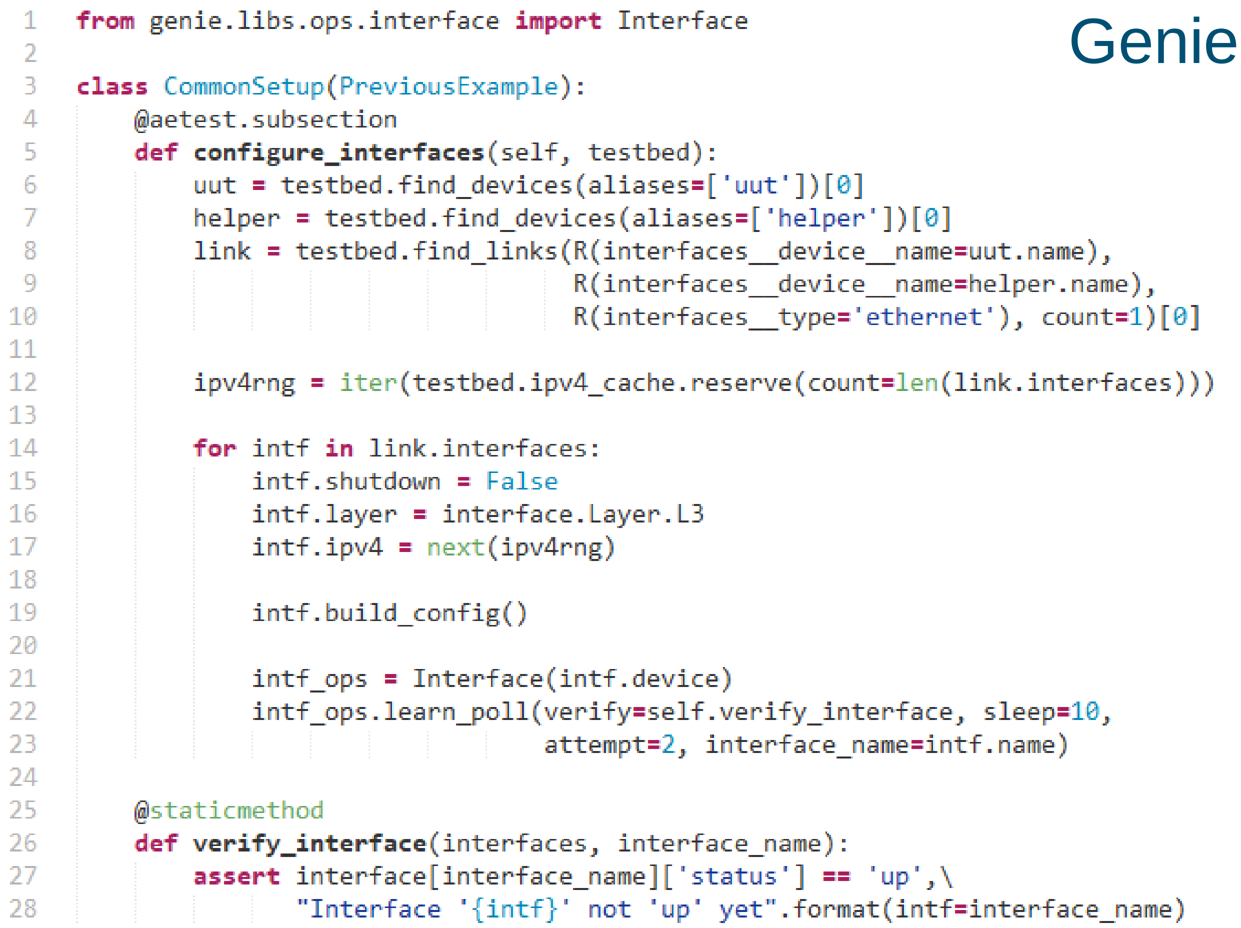
- Use only the classes you need
- SDK's triggers and verifications plug directly into pyATS as test cases and sections

Genie is ***extensible***

- Inherent & extend whenever needed
- Modify only what's required & accommodate for deltas between release/image/etc.

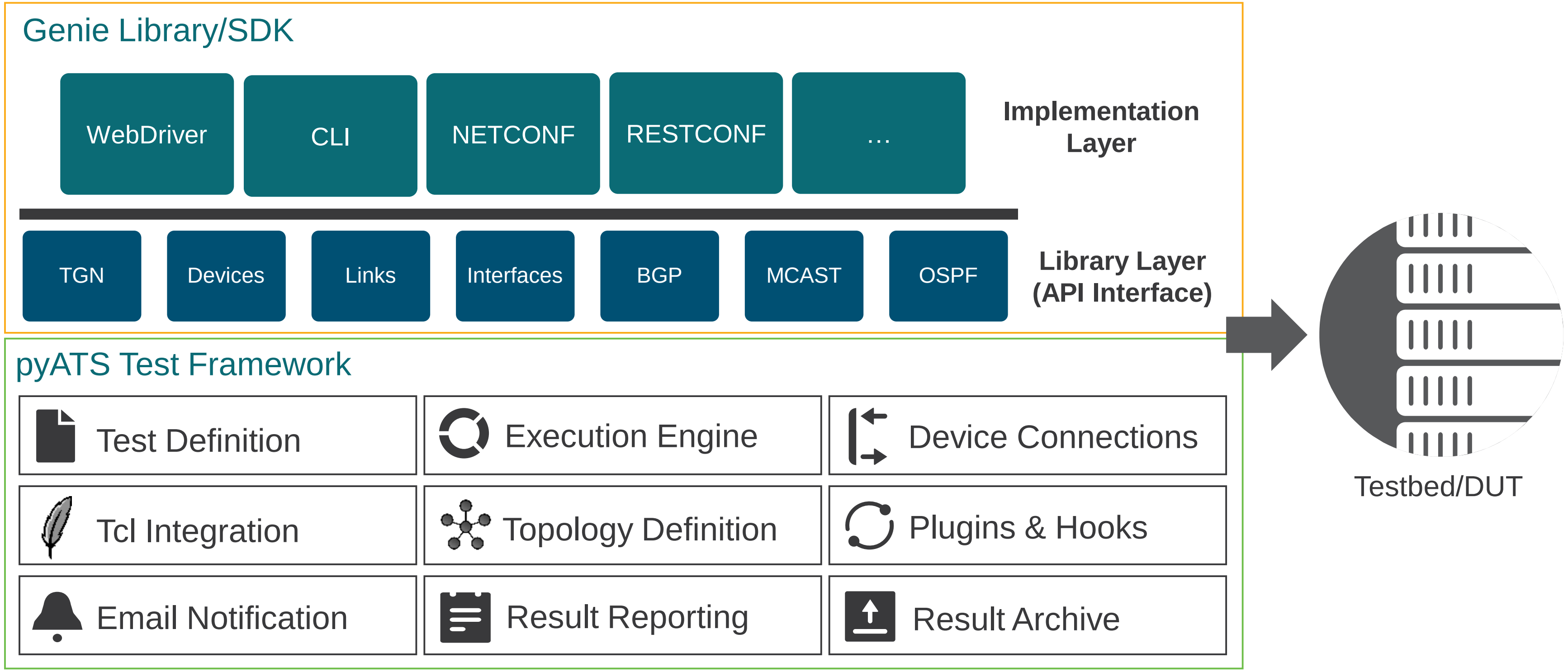
`genie.conf` `genie.ops` `genie.sdk`

Genie: Next Level Testing



```python
from genie.libs.ops.interface import Interface

class CommonSetup(PreviousExample):
    @aetest.subsection
    def configure_interfaces(self, testbed):
        uut = testbed.find_devices(aliases=['uut'])[0]
        helper = testbed.find_devices(aliases=['helper'])[0]
        link = testbed.find_links(R(interfaces__device__name=uut.name),
                                  R(interfaces__device__name=helper.name),
                                  R(interfaces__type='ethernet'), count=1)[0]

        ipv4rng = iter(testbed.ipv4_cache.reserve(count=len(link.interfaces)))

        for intf in link.interfaces:
            intf.shutdown = False
            intf.layer = interface.Layer.L3
            intf.ipv4 = next(ipv4rng)

            intf.build_config()

            intf_ops = Interface(intf.device)
            intf_ops.learn_poll(verify=self.verify_interface, sleep=10,
                                attempt=2, interface_name=intf.name)

    @staticmethod
    def verify_interface(interfaces, interface_name):
        assert interface[interface_name]['status'] == 'up',\
               "Interface '{intf}' not 'up' yet".format(intf=interface_name)
```

Test Ecosystem: pyATS + Genie



Genie SDK

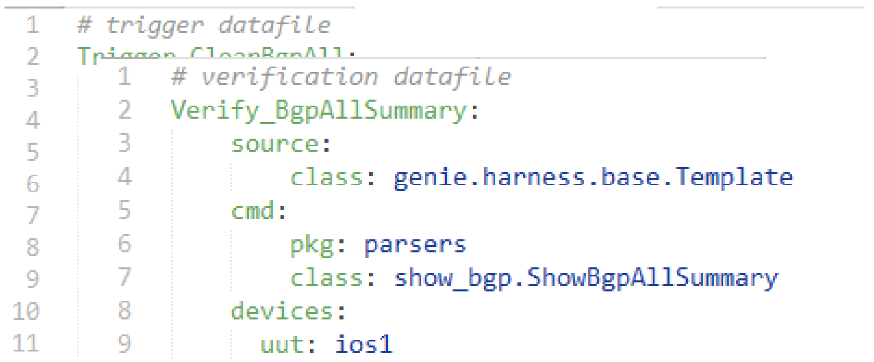
Stimulus & Event Driven

- Pool of triggers & verification: trigger events, verify the aftermath
- Dynamic topology & feature discovery via system/testbed profiling
- Abstraction-enabled: works across a variety of platforms and Cisco IOx platforms.

Agnostic, Data driven Tests

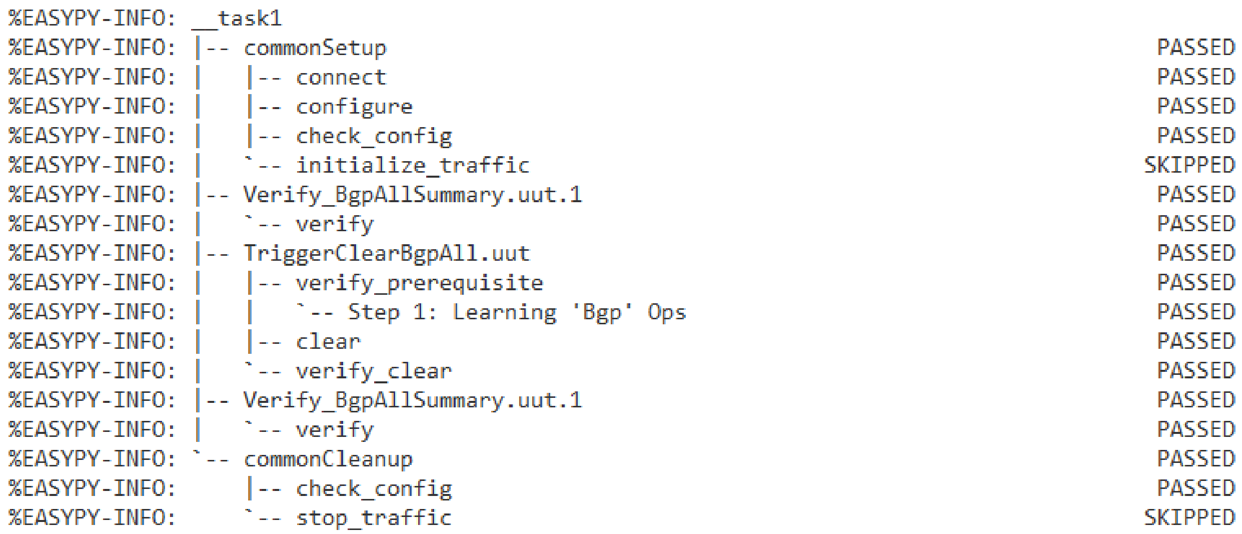
- Plug & Play: select test scenarios based on component and required feature coverage
- Dynamic: runtime generation of testcases based on input data (yaml) file, combining triggers & verifications as per demand
- Reusable: plugs directly into any existing pyATS test script

Reusable, Scalable Test Automation



```
# trigger datafile
Trigger_ClearBgpAll:
```

```
# verification datafile
Verify_BgpAllSummary:
    source:
        class: genie.harness.base.Template
    cmd:
        pkg: parsers
        class: show_bgp.ShowBgpAllSummary
    devices:
      uut: ios1
```



```
%EASYPY-INFO: __task1
%EASYPY-INFO: |-- commonSetup                                              PASSED
%EASYPY-INFO: |   |-- connect                                              PASSED
%EASYPY-INFO: |   |-- configure                                            PASSED
%EASYPY-INFO: |   |-- check_config                                         PASSED
%EASYPY-INFO: |   `-- initialize_traffic                                  SKIPPED
%EASYPY-INFO: |-- Verify_BgpAllSummary.uut.1                               PASSED
%EASYPY-INFO: |   `-- verify                                               PASSED
%EASYPY-INFO: |-- TriggerClearBgpAll.uut                                   PASSED
%EASYPY-INFO: |   |-- verify_prerequisite                                  PASSED
%EASYPY-INFO: |   |   `-- Step 1: Learning 'Bgp' Ops                       PASSED
%EASYPY-INFO: |   |-- clear                                                PASSED
%EASYPY-INFO: |   `-- verify_clear                                         PASSED
%EASYPY-INFO: |-- Verify_BgpAllSummary.uut.1                               PASSED
%EASYPY-INFO: |   `-- verify                                               PASSED
%EASYPY-INFO: `-- commonCleanup                                            PASSED
%EASYPY-INFO:     |-- check_config                                         PASSED
%EASYPY-INFO:     `-- stop_traffic                                        SKIPPED
```

Robot Framework Integration

< 200 LoC

```
*** Settings ***
Resource        rasta.robot
Library         pyats.pyATS

*** Variables ***
${datafile}    datafile.yaml
${testbed}     testbed.yaml

*** Test Cases ***
# Rasta - Example
Connect
    use testbed "${testbed}"
    connect session "vty" via "vty" to device "R1"

Send command on R1
    execute command "show bgp all" on session "vty" on device "R1"

# pyATS - Example
CommonSetup
    execute testcase basic_example_script.common_setup    extra_arg=5

Testcase_One
    execute testcase basic_examples.basic_example_script.tc_one    extra_arg=bgp

parser show bgp all detail
    ${output}=    parse on session "vty" on device "R1" using "parser.show_bgp.ShowBgpAllDetail" with context "cli"
    Log To Console    ${output}

Learn bgp
    ${output}=    learn "bgp" on session "vty" on device "R1" with context "cli"
    Log To Console    ${output}
```

Getting Started

pyATS is available in the Python Package Index (PyPI)

- https://pypi.python.org/pypi/pyats/

Requirements:

- Linux Environment (including WSL)
- Python 3.4.x virtual environment

```
# create a new python virtual environment
$ python3 –m venv ~/pyats

# install in your new environment
$ source ~/pyats/bin/activate
$ pip install pyats genie
```

Examples are available under `~/pyats/examples` after installation.

Day Zero Packages

Genie.Abstract

- Standardizes platform-agnostic library definition and structure
- Dynamic function lookup with auto-fallback based on current tokens

Genie.Metaparser

- Promotes easy-to-maintain parser library structure
- Structure/schema unification among different (but similar) output contexts

Genie.WebDriver

- Selenium web page object design pattern on steroids
- Integrates with pyATS models (testbed YAML, connection classes)

Genie.Robot

- Calling pyATS data structures, libraries and test cases in Robot
- Reusing Genie libraries and SDK in Robot

Unicon

- Universal CLI Connection class: telnet, ssh to network devices
- Platform independent core: new platform support via plugins

Parsergen

- Automated CLI table to data structure converter
- CLI parser using markup language instead of regular expressions

Upcoming Releases

Genie.Telemetry

- Provide a generic, plugin-based telemetry infrastructure to collect statistical and analytical data from your testbed devices

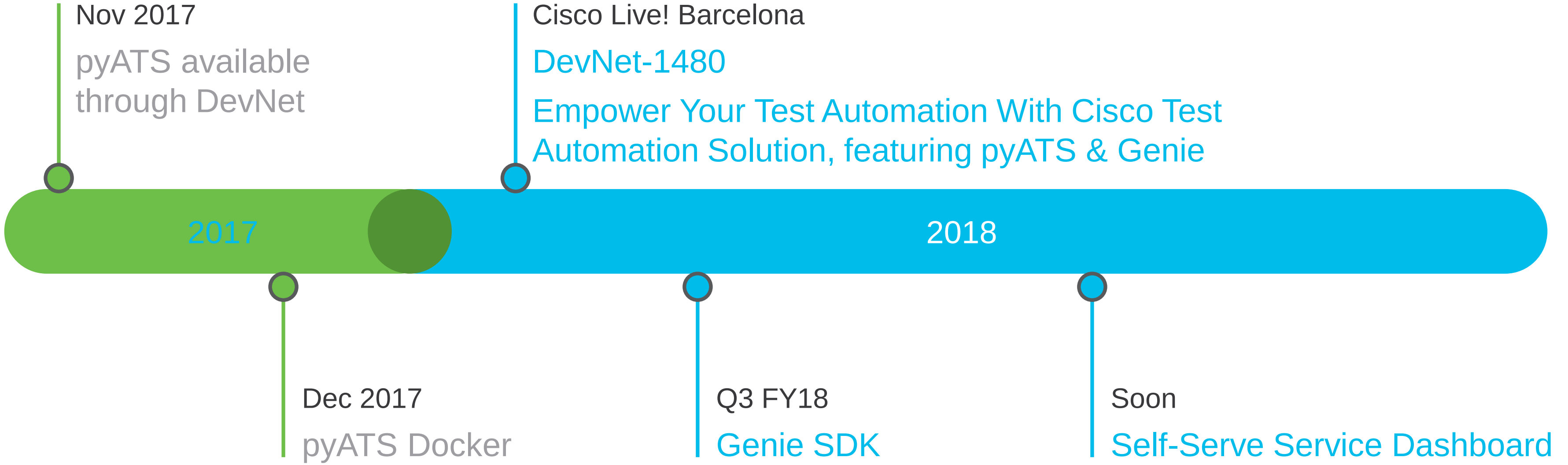
YANG.Connector

- NcClient adaptor to pyATS connection model
- NETCONF configuration tools: merge, diff

REST.Connector

- Request package adapter to pyATS connection model

Roadmap



Nov 2017

pyATS available through DevNet

Cisco Live! Barcelona

DevNet-1480

Empower Your Test Automation With Cisco Test Automation Solution, featuring pyATS & Genie

Dec 2017

pyATS Docker Image Release

Q3 FY18

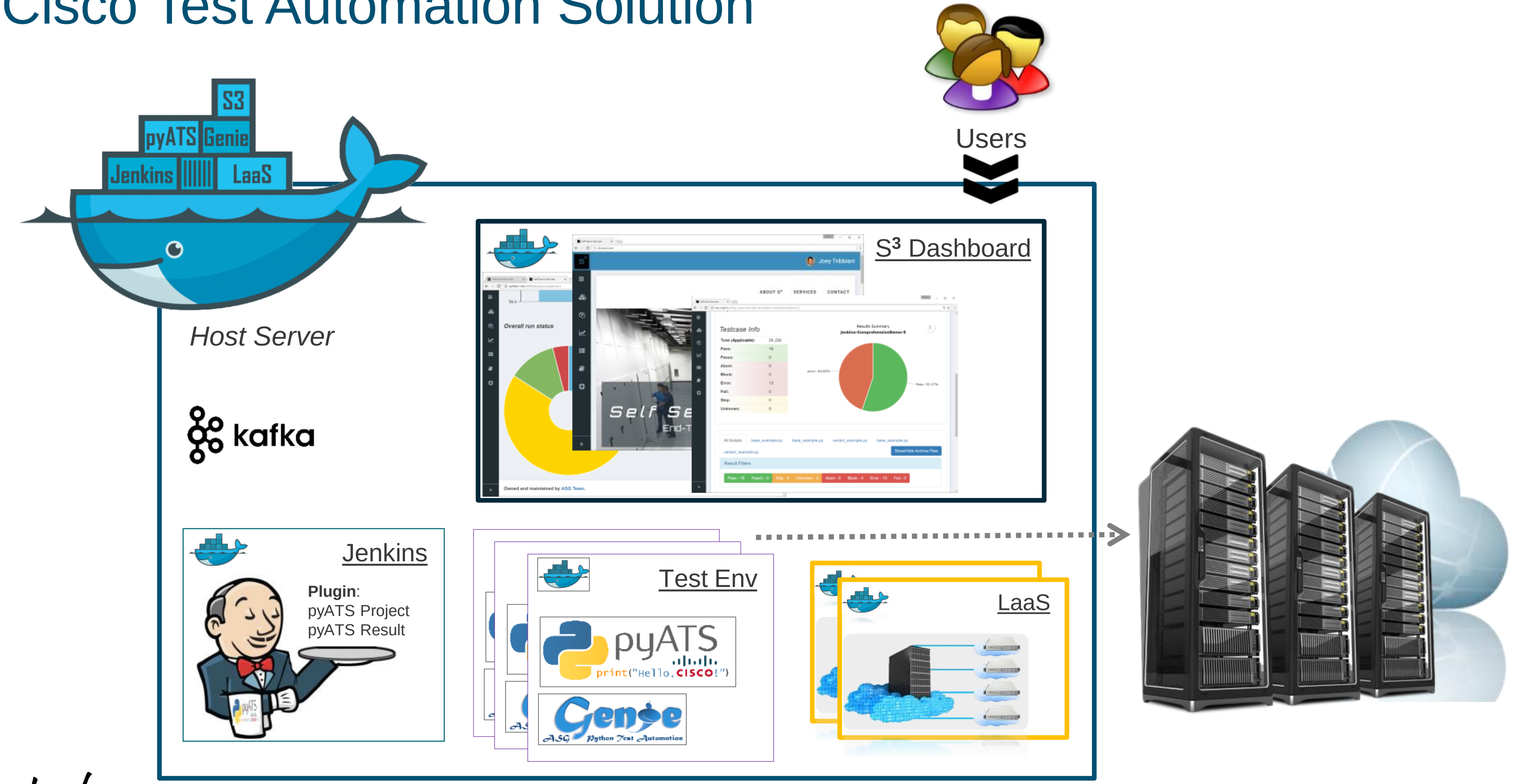
Genie SDK Released on DevNet,

Integration with Robot Framework

Soon

Self-Serve Service Dashboard for pyATS/Genie test suites, featuring testbed management, reservation, monitoring, etc.

Cisco Test Automation Solution



Resources

- DevNet: pyATS - https://developer.cisco.com/site/pyats/
- Framework Documentation: https://developer.cisco.com/site/pyats/docs/
- Package Documentation: https://developer.cisco.com/site/pyats/docs/packages/
- Community Forum: https://communities.cisco.com/community/developer/pyats
- GitHub Folder: https://github.com/CiscoTestAutomation
- DockerHub: https://hub.docker.com/r/ciscotestautomation/pyats/

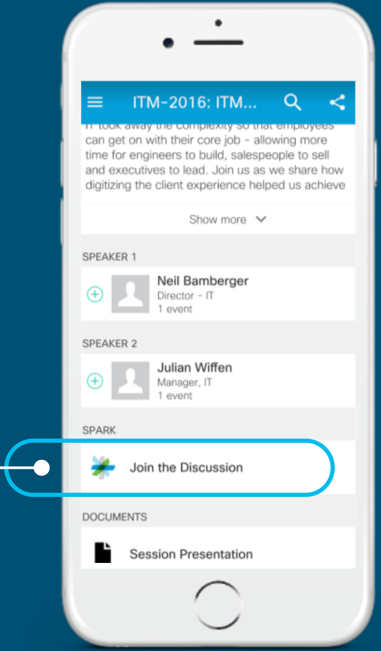
Cisco Spark

Questions?

Use Cisco Spark to communicate with the speaker after the session

How

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



ITM-2016: ITM...

can get on with their core job - allowing more time for engineers to build, salespeople to sell and executives to lead. Join us as we share how digitizing the client experience helped us achieve

cs.co/ciscolivebot#DEVNET-1480

Complete Your Online Session Evaluation

- Please complete your Online Session Evaluations after each session
- Complete 4 Session Evaluations & the Overall Conference Evaluation (available from Thursday) to receive your Cisco Live T-shirt
- All surveys can be completed via the Cisco Live Mobile App or the Communication Stations

Don’t forget: Cisco Live sessions will be available for viewing on-demand after the event at www.ciscolive.com/global/on-demand-library/.

Continue Your Education

- Demos in the Cisco campus
- Walk-in Self-Paced Labs
- Tech Circle
- Meet the Engineer 1:1 meetings
- Related sessions

Thank you

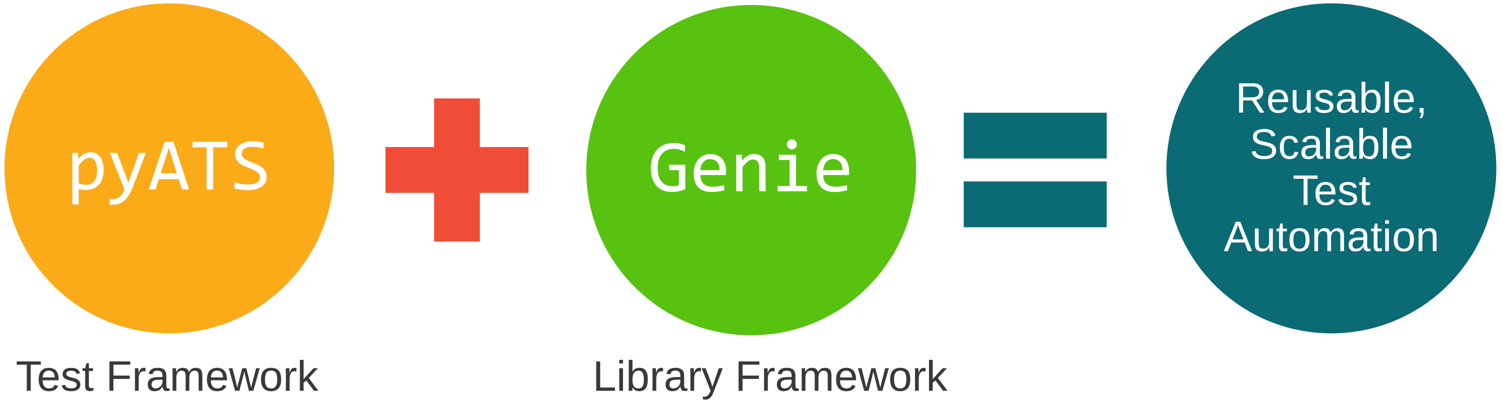
You're

iT

Cisco live!

Backups

Test Ecosystem


pyATS
Genie
Reusable, Scalable Test Automation
Test Framework
Library Framework