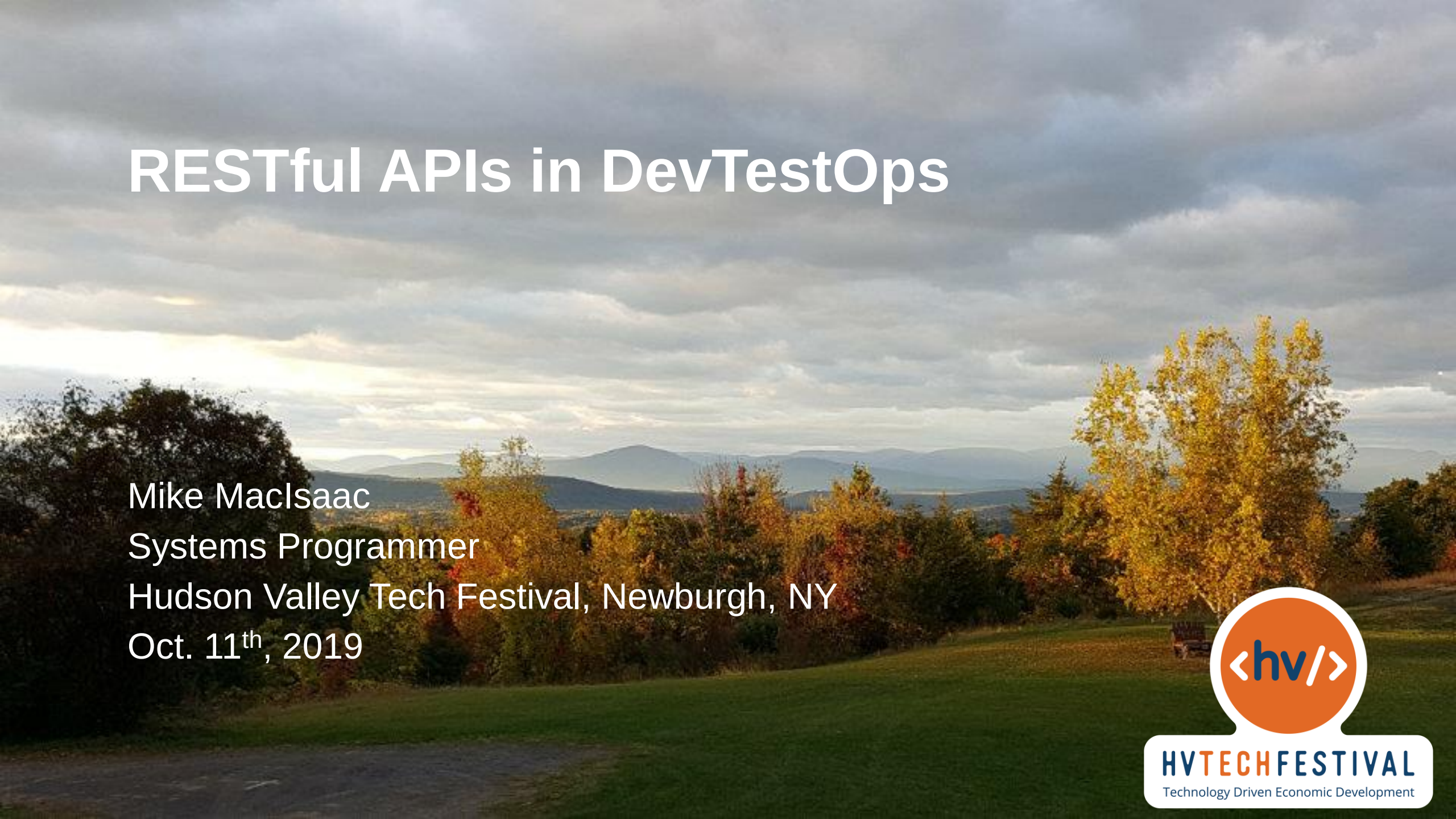
RESTful APIs in DevTestOps

Mike MacIsaac
Systems Programmer
Hudson Valley Tech Festival, Newburgh, NY
Oct. 11th, 2019



Abstract

We'll discuss how developing RESTful APIs enables services from your team's resources to be utilized by other teams while also allowing for automated testing of the services you provide.

The speaker has been working on an in-house private cloud solution where Linux virtual machines running on the mainframe are made available to application development and solution teams. The solution offers 'self-service' to the end-user teams and allows for a 'single pane of glass' across the enterprise.

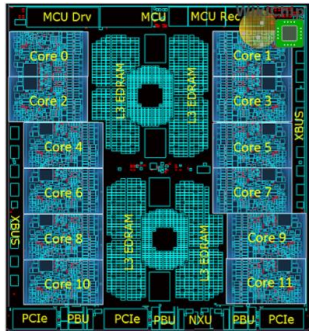
The talk will start with business problems and solutions. What does this solve? Then some detail of the RESTful API interface and it's usefulness in testing will be presented.

Agenda

- Wait, Mainframes are still around?
- Private Cloud on mainframes
- Business needs and payback
- DevTestOps
- RESTful APIs

Wait, Mainframes are still around?

- Absolutely
- The new z15
  - Most highly available computer hardware (MTBF > 20 years)
  - Up to 40 TB memory
  - Up to 190 CPUs at 5.2 GHz
  - Core has 12 CPUs, massive cache, 9 Billion transistors
  - I/O to DASD, SAN
  - Pervasive encryption

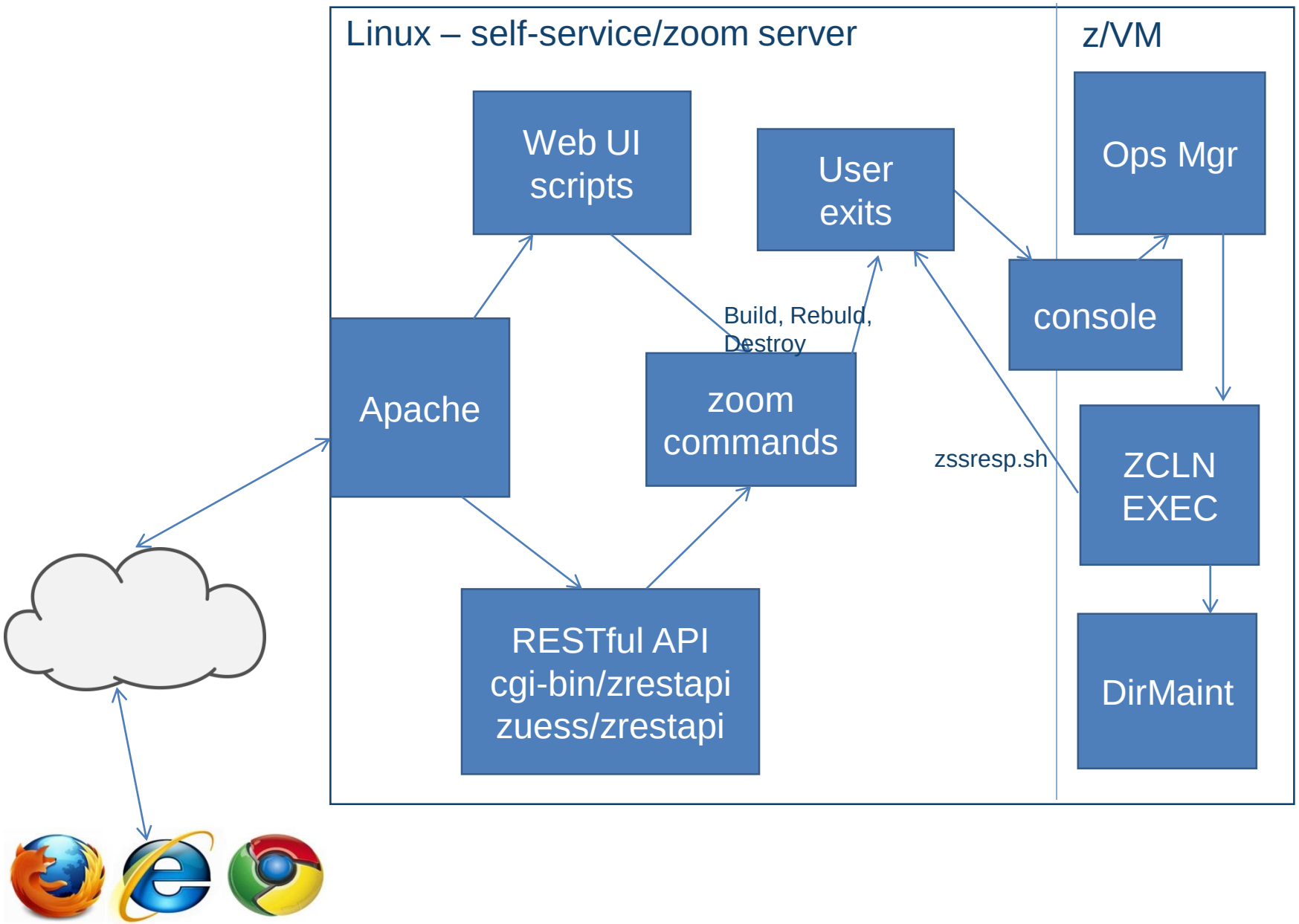


Private cloud on mainframes

- Requirements
  - Allow self-service for associates
    - Build, configure, manage, remove zLinux virtual servers
    - Browser-based user interface
    - RESTful API
  - Authentication - LDAP credentials
  - Enforce authorization
    - By LDAP groups
    - Members see only their group's systems
    - Admins can see all systems
  - Enable a two-tier grouping mechanism
    - Primary: LDAP groups
    - Secondary: Arbitrary grouping called projects
  - Allocate quotas by group for CPUs and memory
  - Maintain an audit trail
    - Log user and current group for each operation
  - Allow server filtering:
    - Host name
    - Group:project
    - Other data and metadata
  - Maintain creation and expiration dates and owners
  - Perform locking so only one user can operate on a server
  - Enable user preferences
  - Create reports

Private cloud on mainframes (cont'd)

- Private cloud for zLinux virtual servers
  - Build operations – Build, Rebuild, Destroy, Purge
  - Configure operations – add/remove CPUs/memory/disk
  - Power operations – Reboot, Power on, Power off
- zLinux-centric, z/VM when needed
  - Database is Linux file system – "the tree"
  - Leverages TCP/IP, ssh, scp, Linux file system, rich Linux toolset
  - High Availability: two copies of data with hot standby
- Three access methods
  - Line commands (SSH) – zoom
  - Web UI presentation layer – zuess
  - RESTful APIs – thin layer between Apache and zoom line commands



Business needs and payback

- Business needs
  - Fewer “tickets”
  - Easier access to z resources
- Business solution: “self-service portal” (aka Private cloud)
  - Other vendor products did not perform adequately
  - Write it in-house starting June 2016
- Business payback
  - Application team agility and independence
  - Fewer z system administrators
  - Solid audit trail, job log
- We have not yet reached Nirvana

DevOps

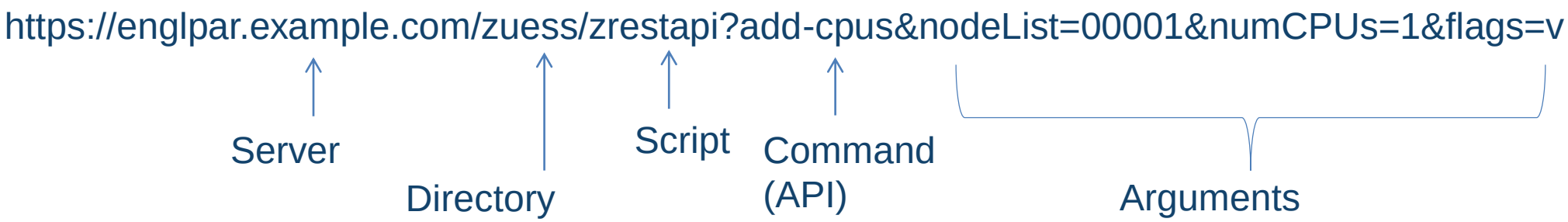
- The evolution of "DevOps"
  - Coined in 2009, DevOps is "development" and "operations"
  - Dev**Sec**Ops is a `practice that rose from DevOps that includes information technology security as a fundamental aspect in all the stages of software development`
  - Dev**Test**Ops - The three most important aspects of DevOps are Testing, Testing and Testing.
- System administrators (Ops) must become developers (Dev)
- New requirements for operations
  - Source code control system
  - Check-in, check-out, forking, rollback, etc.
  - Build, deploy, install and verify
- Dev1/Dev2/QA/Prod * 3+ LPARs per cluster

DevTestOps

- Web UI test driven by Selenium
- Regression test driven by RESTful API calls
- zSS Regression test history
  - 1.0 - early 2017 - different tests built in "Dev 1", "Dev 2" and QA
  - 2.0 - Mike's approach to make tests portable among environments
  - 2.1 – Developer 2’s approach – self-generating JSON input

RESTful APIs – first try

- Simple approach – think Remote Procedure Calls (RPCs)
- Sample URL:



RESTful API def'n file

API command	type	zoom command	Args [opt'l],req'd	Description
help_rest_api	ro	zhelprestapi	[flags]	Show summary of all RESTful APIs
help_zoom_cli	ro	zhelpcli	[flags]	Show summary of all line commands
find_objects	ro	zfindobjects	[flags],pattern	Find objects in zoom tree
list_cecs	ro	zlscecs	[flags],[pattern]	List CECs in the zoom tree
list_lpars	ro	zlslpars	[flags],[pattern]	List LPARs in the zoom tree
list_cpus	ro	zlscpus	[flags],[nodeList]	List CPUs on Linux nodes
list_groups	ro	zlsgroups	[flags],[nodeList]	List zoom groups
list_memory	ro	zlsmemory	[flags],[nodeList]	List memory on Linux nodes
list_minidisks	ro	zlsminidisks	[flags],[nodeList]	List minidisks on Linux nodes
list_systems	ro	zlsclients	[flags],[nodeList]	List Linux nodes
list_commands	ro	zlscommands	[flags]	Show all zoom commands
list_tree	ro	zlstree,-a	[flags]	Show the zoom tree
list_dirs	ro	zls,-a	[flags]	Show directories in the zoom tree
list_quotas	ro	zlsquotas	[flags],[pattern]	Show quotas
ping_systems	ro	zpingnodes,-c	[flags],[nodeList]	Ping Linux systems
query_chpids	ro	zqchpid	[flags],[chanPath]	Query z/VM channel paths
query_dasd	ro	zqdasd	[flags],[rdevRange]	Query z/VM DASD
query_direntry	ro	zqdirentry	[flags],entry	Query a z/VM USER, PROFILE IDENTITY
query_fcp	ro	zqfcp	[flags],[rdevRange]	Query z/VM FCP devices
query_osa	ro	zqosa	[flags]	Query z/VM OSA devices
query_pav	ro	zqpav	[flags],[rdevRange]	Query z/VM PAV devices
query_rdev	ro	zqrdev	[flags],[rdevRange]	Query z/VM real devices

RESTful API def'n file (cont'd)

API command	type	zoom command	Args [opt'l],req'd	Description
help_rest_api	ro	zhelprestapi	[flags]	Show summary of all RESTful APIs
help_zoom_cli	ro	zhelpcli	[flags]	Show summary of all line commands
find_objects	ro	zfindobjects	[flags],pattern	Find objects in zoom tree
list_cecs	ro	zlscecs	[flags],[pattern]	List CECs in the zoom tree
list_lpars	ro	zlslpars	[flags],[pattern]	List LPARs in the zoom tree
list_cpus	ro	zlscpus	[flags],[nodeList]	List CPUs on Linux nodes
list_groups	ro	zlsgroups	[flags],[nodeList]	List zoom groups
list_memory	ro	zlsmemory	[flags],[nodeList]	List memory on Linux nodes
list_minidisks	ro	zlsminidisks	[flags],[nodeList]	List minidisks on Linux nodes
list_systems	ro	zlsclients	[flags],[nodeList]	List Linux nodes
list_commands	ro	zlscommands	[flags]	Show all zoom commands
list_tree	ro	zlstree,-a	[flags]	Show the zoom tree
list_dirs	ro	zls,-a	[flags]	Show directories in the zoom tree
list_quotas	ro	zlsquotas	[flags],[pattern]	Show quotas
ping_systems	ro	zpingnodes,-c	[flags],[nodeList]	Ping Linux systems
query_chpids	ro	zqchpid	[flags],[chanPath]	Query z/VM channel paths
query_dasd	ro	zqdasd	[flags],[rdevRange]	Query z/VM DASD
query_direntry	ro	zqdirentry	[flags],entry	Query a z/VM USER, PROFILE IDENTITY
query_fcp	ro	zqfcp	[flags],[rdevRange]	Query z/VM FCP devices
query_osa	ro	zqosa	[flags]	Query z/VM OSA devices
query_pav	ro	zqpav	[flags],[rdevRange]	Query z/VM PAV devices
query_rdev	ro	zqrdev	[flags],[rdevRange]	Query z/VM real devices

RESTful API - current

- More descriptive hierarchy in URL with ‘endpoints’
- JSON input and output
- Operations endpoints
  - operations/**build**/
    - Many required and optional arguments
  - operations/**rebuild**/
  - operations/**destroy**/
  - operations/**configure**/
    - requestedFor - user performing the operation
    - hostName=name|zss_uuid=UUID - system to be configured
    - quantity=positiveInteger|negativeInteger
    - quantityType=relative|absolute
    - resourceType=CPUs|memory|minidisk
  - operations/**power**/
    - requestedFor - user performing the operation
    - hostName=name|zss_uuid=UUID - system to be powered
    - operationType=powerOn|powerOff|rebootHard|rebootSoft
  - operations/**restore**/ - bring a server back from ‘death row’

RESTful API - current

- List endpoints
  - list/bptid/<filter> List one or more BPT IDs in the mapping file
  - list/jobs/<filter> List contents of one or more jobs
  - list/jobnames/<filter> List name of one or more jobs
  - list/systems/bptid/<filter> List systems by BPT ID
  - list/systems/hostname/<filter> List system(s) by host name pattern
  - list/systems/uuid/<filter> List one system by zss UUID
  - list/version List current version of zSS code
- Monitor endpoint
  - **monitor**/jobs/<filter>

Main input file or “test bucket”

```
# head -15 regrtest2.input
# list by hostname, UUID, job ID and zSS version
list      hostname/HOSTNAME0
list      uuid/UUID0
list      job/JOB0
list      version
# build SYSTEM1 monitor and verify system is up
build     build.SYSTEM1.JSON
monitor   job-00001.build
verify    SYSTEM1 isUp
# set SYSTEM1 CPUs to 4, monitor and verify
configure setCPUs4.SYSTEM1.JSON
monitor   job-00001.setCPUs4
verify    SYSTEM1 CPUs 4
```

Sample JSON input file

```
{ "requestedFor": "macisaam@es.ad.adp.com",
 "flags": "FLAGS",
 "systems":
  [ { "hostName": "SYSTEM1",
      "operationType": "build",
      "CPUs": "3",
      "distro": "SLES12",
      "env": "MikesEnvironment",
      "group": "vmlinux",
      "jobID" : "job-00001.build",
      "memory": "2G",
      "serverType": "MikesServerType",
      "systemType": "MikesSystemType",
      "userID": "USERID1"
    }  ]  }
```

References

- Zoom and zuess RPMs
  - https://sourceforge.net/projects/system-zoom/
- ***The DEVOPS Handbook***
  - https://itrevolution.com/book/the-devops-handbook/
- Mike's website
  - https://sites.google.com/site/mike99mac/home

Questions?

OpenHub
Innovation | Collaboration | Education

SUNY ORANGE

Mount Saint Mary College

ORANGE COUNTY NEW YORK
1 NOVEMBER 1683

OpenHub
Innovation | Collaboration | Education

SUNY ORANGE

Mount Saint Mary College

ORANGE COUNTY
NEW YORK
1 NOVEMBER 1683