GSE | GUIDE SHARE EUROPE

UK Region

See Ansible Automation Platform in action, provisioning z/OS middleware with the latest CICS TS collection

Kiera Bennett and Andrew Hughes
IBM UK Ltd

Session EK

GSE UK Conference
2024 Charities

Every year, speakers and delegates at the GSE UK Conference donate to the charities we help. This year is no exception and we aim to collect donations for two worthy causes:
Air Ambulance UK and Muscular Dystrophy UK.

Please consider showing your appreciation by kindly donating a small sum to our charities this year!!

AIR AMBULANCES UK
SUPPORTING AIR AMBULANCE CHARITIES

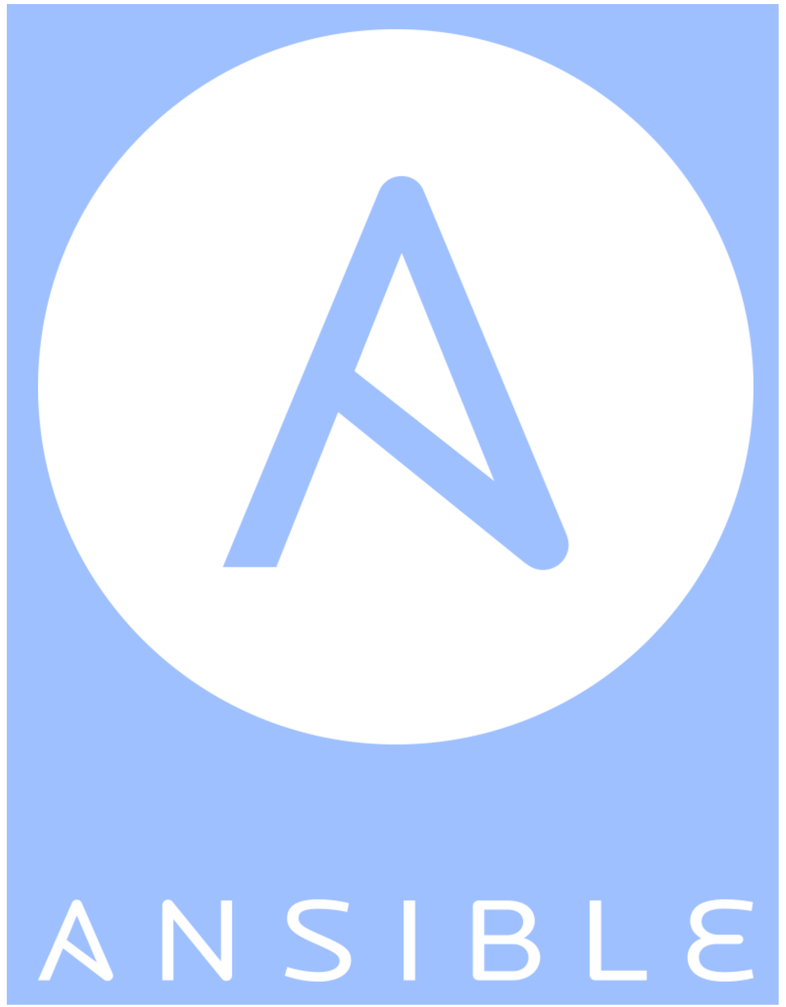
MUSCULAR DYSTROPHY UK | OUR MUSCLES MATTER

What is Ansible?

Ansible is a provisioning, configuration management and application deployment tool.

Tagline: “Turn tough tasks into repeatable playbooks”.

Rather than managing one system at a time, Ansible models your IT infrastructure by describing how all your systems inter-relate.



Common usage of Ansible

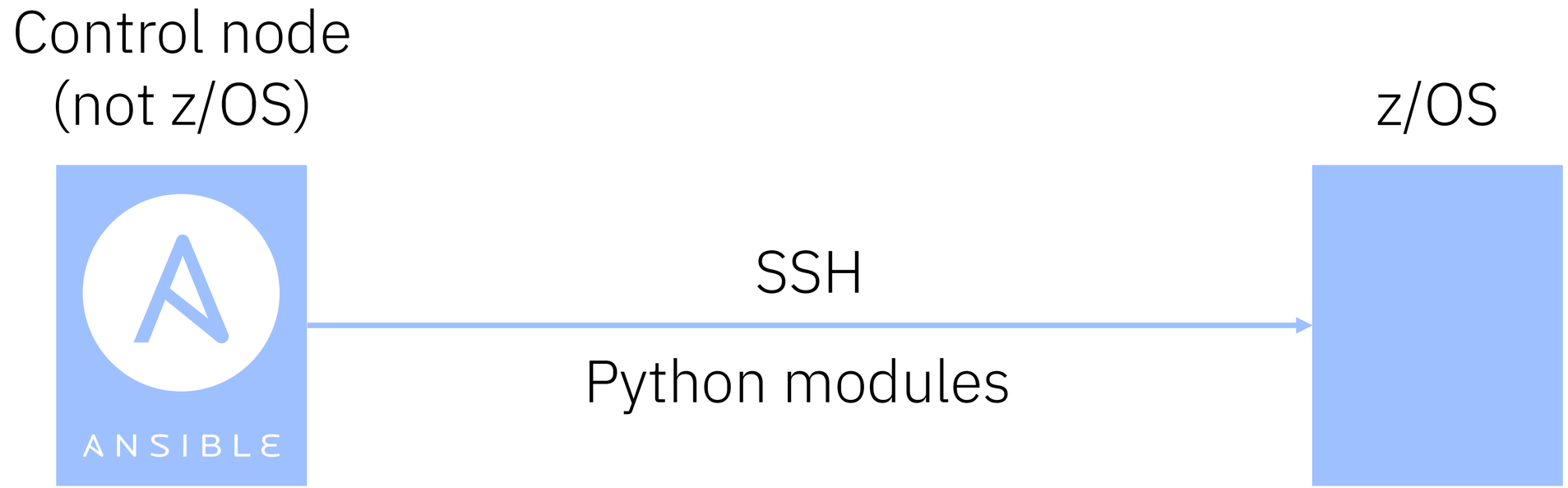
- System provisioning
- Installing applications
- Managing users
- Updating certificates
- Continuous delivery
- Configuration management

Why is Ansible so popular?

Ansible is extremely popular across many enterprises. For example, it’s now the top cloud configuration tool and is heavily used on-prem too.

- Generically applicable to lots of scenarios
- Easy to write automation
- Opportunity to standardize enterprise automation strategy
- Makes configuration as code more realistic
- Thousands of community developed extensions

Ansible architecture

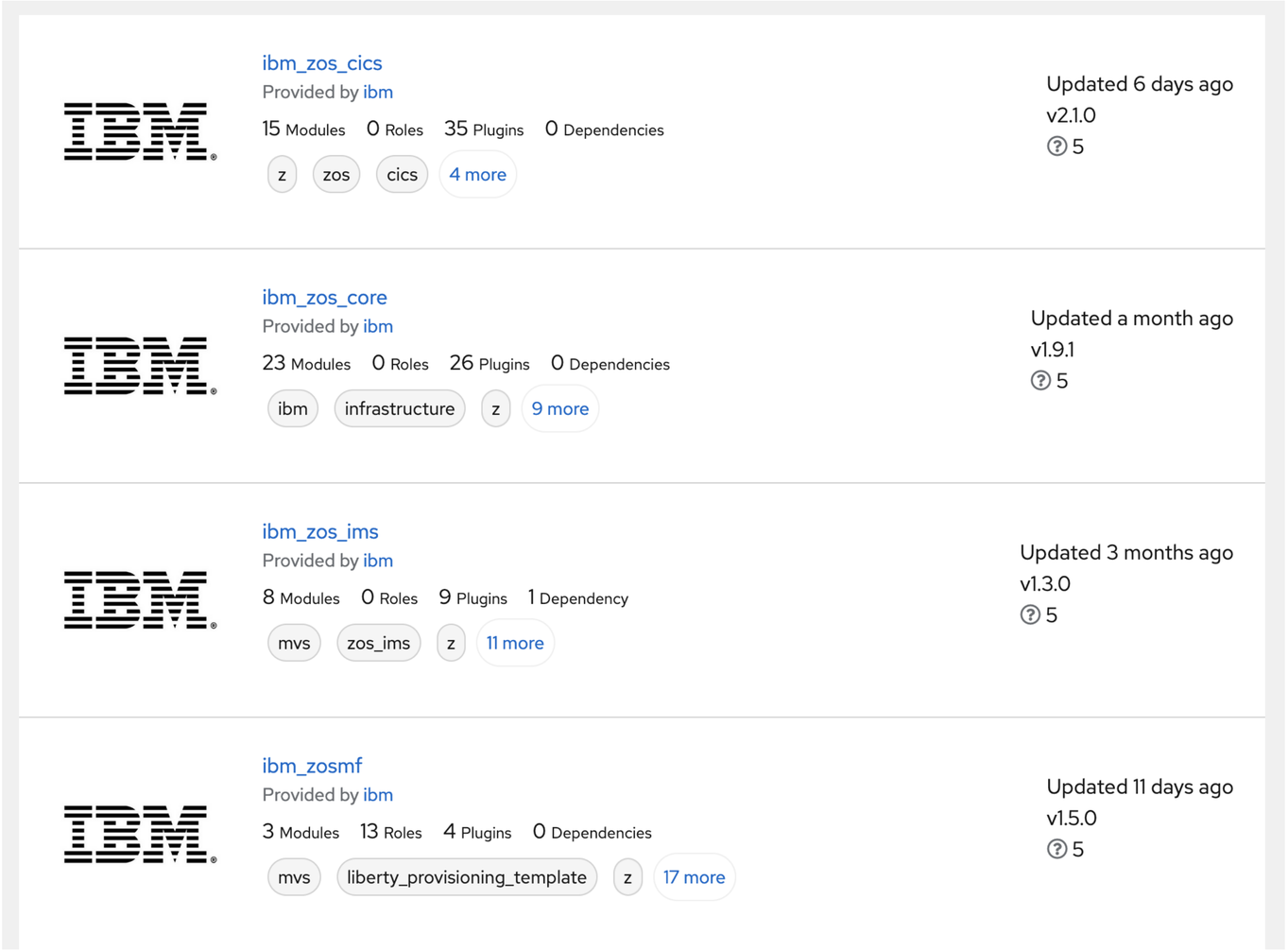


Ansible z/OS Ecosystem

There is a growing set of collections that support z/OS function on Ansible Galaxy -> the repository for Ansible collections of tasks.

In addition, they're also part of the Red Hat Ansible Certified Content for IBM Z. This means they're available on Ansible Automation Platform and are fully supported by Red Hat and IBM.

The ibm_zos_core collection gives a good foundational experience on z/OS around datasets, APF, transferring data, operator commands, and so on.



CICS Collection for Ansible

CICS collection for Ansible provides you with tasks that are commonly required when working with CICS TS.

It’s being developed in the open at github.com/ansible-collections/ibm_zos_cics

The collection is in Ansible Galaxy at galaxy.ansible.com/ibm/ibm_zos_cics

The collection is part of the Red Hat Ansible Certified Content for IBM Z.

We’d like to show you a little about how it works with some demo scenarios

Modules (15)

- aux_temp_storage
- aux_trace
- cmci_action
- cmci_create
- cmci_delete
- cmci_get
- cmci_update
- csd
- global_catalog
- local_catalog
- local_request_queue
- region_jcl
- stop_region
- td_intrapartition
- transaction_dump

Modules in the CICS collection <2.1.0

The CICS collection includes the following tasks:

- cmci_action
- cmci_create
- cmci_delete
- cmci_get
- cmci_update

All use CMCI (the HTTP API to manage CICS) so can be run on the target node (the z/OS LPAR) or remotely.
CMCI is the same API that Explorer uses. You can use our Ansible collection to automate lots of things you might do manually today in Explorer.

Scenario

A regular need as a CICS system programmer is to capture the state of CICS resources, perhaps to compile the data into a report.

To some extent this is possible through CICS Explorer. But Ansible could provide a more automated and less error-prone way of doing it...

CMCI Reporting Sample

```yaml
##############################################################################
# Get information about CICS regions from the supplied context, using CMCI
##############################################################################
- name: Get information about CICS regions
  ibm.ibm_zos_cics.cmci_get:
    context: "{{ context }}"
    cmci_host: "{{ cmci_host }}"
    cmci_port: "{{ cmci_port | int }}"
    cmci_user: "{{ cmci_user | default(omit) }}"
    cmci_password: "{{ cmci_password | default(omit) }}"
    scheme: "{{ scheme }}"
    type: "CICSRegion"
  register: result
```

```
reporting ansible-playbook report.yml
Target CMCI hostname: winmvs28.hursley.ibm.com
Target CMCI port: 28963
CMCI scheme [https]: http
Target CPSM context: CICSEX61
CMCI user name (leave blank for unauthenticated):
CMCI password (leave blank for unauthenticated):
```

	A	B	C	D
1	eyu_cicsnam	release	jobid	totltasks
2	IYCWENK1	0710	STC34213	125
3	IYCWENL1	0720	STC34212	128
4	IYCWENM1	0730	STC34211	126
5	IYCWENN1	0740	STC34210	129
6	IYCWENTH	0740	STC34214	189
7	IYCWENW1	0740	STC34208	130
8	IYCWENW2	0740	STC34209	341
9				

New modules in the CICS collection 2.1.0+

aux_temp_storage	Create and remove the CICS auxiliary temporary storage data set
aux_trace	Allocate auxiliary trace data sets
csd	Create, remove, and manage the CICS CSD
global_catalog	Create, remove, and manage the CICS global catalog
local_catalog	Create, remove, and manage the CICS local catalog
local_request_queue	Create and remove the CICS local request queue
region_jcl	Create CICS startup JCL data set
stop_region	Stop a CICS region
td_intrapartition	Create and remove the CICS transient data intrapartition data set
transaction_dump	Allocate transaction dump data sets

Why did the version number change?

We removed support for Python 2 (on the controller, was never supported on z/OS)

The new modules also depend on ibm_zos_core 1.5.0+, you must have that installed in the control node

Consequentially, you'll also need a compatible version of ZOAU installed on z/OS

IBM Open Enterprise Python + ZOAU Support

With z/OS 3.1, IBM Open Enterprise Python and ZOAU are now available to install with z/OS as no-charge products

They’re available at no-cost license and no-cost S&S

i.e. they’re bundled and supported with z/OS 3.1

See the announcement

New modules in the CICS collection 2.1.0+

aux_temp_storage	Create and remove the CICS auxiliary temporary storage data set
aux_trace	Allocate auxiliary trace data sets
csd	Create, remove, and manage the CICS CSD
global_catalog	Create, remove, and manage the CICS global catalog
local_catalog	Create, remove, and manage the CICS local catalog
local_request_queue	Create and remove the CICS local request queue
region_jcl	Create CICS startup JCL data set
stop_region	Stop a CICS region
td_intrapartition	Create and remove the CICS transient data intrapartition data set
transaction_dump	Allocate transaction dump data sets

csd

You can allocate and initialize a new CICS csd with a task configured something like this

The module will run DFHCSDUP to initialize the new CSD

```
tasks:
  - name: Allocate and initialize a new CSD
    ibm.ibm_zos_cics.csd:
      cics_data_sets:
        sdfhload: CTS610.CICS740.SDFHLOAD
      region_data_sets:
        dfhcsd:
          dsn: STEWF.RGNS.CSD.DFHCSD
      state: initial      You, 1 second ago • l
```

csd

```
      DEFINE CLUSTER (NAME(STEWF.RGNS.CSD.DFHCSD) -
      MEGABYTES(4 1) -
      RECORDSIZE(200 2000) -
      INDEXED -
      KEYS(22 0) -
      FREESPACE(10 10) -
      SHAREOPTIONS(2) -
      REUSE) -
      DATA (NAME(STEWF.RGNS.CSD.DFHCSD.DATA) -
      CONTROLINTERVALSIZE(8192)) -
      INDEX (NAME(STEWF.RGNS.CSD.DFHCSD.INDEX))
0IDC0508I DATA ALLOCATION STATUS FOR VOLUME P2P843 IS 0
0IDC0509I INDEX ALLOCATION STATUS FOR VOLUME P2P843 IS 0
 IDC0181I STORAGECLASS USED IS STANDARD
 IDC0181I MANAGEMENTCLASS USED IS STANDARD
0IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0
0
```

```
********************************************************************************
**  CICS RDO OFF-LINE UTILITY PROGRAM DFHCSDUP RELEASE:0740 PTF:UI92380 .**
********************************************************************************




INITIALIZE



DFH5120 I PRIMARY CSD OPENED;  DDNAME: DFHCSD   - DSNAME: STEWF.RGNS.CSD.DFHCSD
DFH5280 I PROCESSING DEFINITIONS FROM LIBRARY MEMBER DFHCURDI
DFH5131 I LIST DFHLIST  CREATED.
DFH5135 I GROUP DFHDCTG  ADDED TO LIST DFHLIST
DFH5135 I GROUP DFHBMS   ADDED TO LIST DFHLIST
DFH5135 I GROUP DFHCONS  ADDED TO LIST DFHLIST
DFH5135 I GROUP DFHDBCTL ADDED TO LIST DFHLIST
DFH5135 I GROUP DFHDB2   ADDED TO LIST DFHLIST
DFH5135 I GROUP DFHEDF   ADDED TO LIST DFHLIST
DFH5135 I GROUP DFHEDP   ADDED TO LIST DFHLIST
DFH5135 I GROUP DFHEP    ADDED TO LIST DFHLIST
DFH5135 I GROUP DFHFE    ADDED TO LIST DFHLIST
DFH5135 I GROUP DFHHARDC ADDED TO LIST DFHLIST
DFH5135 I GROUP DFHINQUI ADDED TO LIST DFHLIST
DFH5135 I GROUP DFHINTER ADDED TO LIST DFHLIST
DFH5135 I GROUP DFHISC   ADDED TO LIST DFHLIST
DFH5135 I GROUP DFHMISC  ADDED TO LIST DFHLIST
```

csd

You can also run a DFHCSDUP script by using state: changed.

Here we allocate and initialize a CSD in one task, and run a DFHCSDUP script in a second task

The script can be provided in-line in the playbook, transmitted as a local file, or already on z/OS in a data set or USS file

```yaml
tasks:
  - name: "Allocate and initialize a CSD"
    ibm.ibm_zos_cics.csd:
      cics_data_sets:
        sdfhload: CTS610.CICS740.SDFHLOAD
      region_data_sets:
        dfhcsd:
          dsn: STEWF.RGNS.CSD.DFHCSD2
      state: initial

  - name: |
      Create a group list for my region, and add
      the auto install terminal definitions to it
    ibm.ibm_zos_cics.csd:
      cics_data_sets:
        sdfhload: CTS610.CICS740.SDFHLOAD
      region_data_sets:
        dfhcsd:
          dsn: STEWF.RGNS.CSD.DFHCSD2
      state: changed
      input_location: INLINE
      input_content: |
        ADD GROUP(DFHTERMC) LIST(DFHLIST1)

        LIST LIST(DFHLIST1) OBJECTS
```

csd

```
ADD GROUP(DFHTERMC) LIST(DFHLIST1)


DFH5120 I PRIMARY CSD OPENED;  DDNAME: DFHCSD   - DSNAME: STEWF.RGNS.CSD.DFHCSD2
DFH5131 I LIST DFHLIST1 CREATED.
DFH5135 I GROUP DFHTERMC ADDED TO LIST DFHLIST1
DFH5101 I ADD COMMAND EXECUTED SUCCESSFULLY.
```

```
********************************************************************************************************************************

OBJECTS IN GROUPS IN LIST                                          UTILITY COMMAND:    LIST LIST(DFHLIST1)     OBJECTS

********************************************************************************************************************************

GROUP NAME: DFHTERMC    (IBM PROTECTED)
----------

TERMINAL(AUTC)         GROUP(DFHTERMC) DESCRIPTION()          AUTINSTMODEL(ONLY)     AUTINSTNAME(DFHCONS)   24.220 01:25
                       TERMINAL-IDENTIFIERS
                                       TYPETERM(DFHCONS)      NETNAME(AUTC)          CONSNAME(CONSOLE)
                                       REMOTESYSTEM()         REMOTENAME()           REMOTESYSNET()
                                       MODENAME()
                       ASSOCIATED-PRINTERS
                                       PRINTER()              PRINTERCOPY(NO)        ALTPRINTER()
                                       ALTPRINTCOPY(NO)
                       PIPELINE-PROPERTIES
                                       POOL()                 TASKLIMIT(NO)
                       OPERATOR-DEFAULTSPRESET-SECURITY
                                       USERID(*EVERY)         NATLANG(E)
                       TERMINAL-USAGES
                                       TRANSACTION()          TERMPRIORITY(0)        INSERVICE(YES)
```

csd

We can simply the previous playbook by using module_defaults

The ibm_zos_cics collection defines a module_defaults group which can be used to set common paramers on region provisioning tasks

This way we can avoid duplicating the configuration

```yaml
module_defaults:
  group/ibm.ibm_zos_cics.region:
    cics_data_sets:
      sdfhload: CTS610.CICS740.SDFHLOAD
    region_data_sets:
      dfhcsd: STEWF.RGNS.CSD.DFHCSD2

tasks:
  - name: "Allocate and initialize a CSD"
    ibm.ibm_zos_cics.csd:
      state: initial

  - name: |
      Create a group list for my region, and add
      the auto install terminal definitions to it
    ibm.ibm_zos_cics.csd:
      state: changed
      input_location: INLINE
      input_content: |
        ADD GROUP(DFHTERMC) LIST(DFHLIST1)

        LIST LIST(DFHLIST1) OBJECTS
```

Template expressions

If we add the global catalog task to the same playbook, we can still take advantage of module_defaults

Data set locations can be expressed with a template, using the special handle `<< data_set_name >>` which is substituted for the default name for the module in question (e.g. DFHCSD, DFHGCD)

We can use one template to define the naming conventions for all our data sets, and share it between all modules with module_defaults

See the doc

```yaml
module_defaults:
  group/ibm.ibm_zos_cics.region:
    cics_data_sets:
      sdfhload: CTS610.CICS740.SDFHLOAD
    region_data_sets:
      template: "STEWF.RGNS.DEMO.<< data_set_name >>"

tasks:
  - name: "Allocate and initialize a CSD"
    ibm.ibm_zos_cics.csd:
      state: initial

  - name: |
      Create a group list for my region, and add
      the auto install terminal definitions to it
    ibm.ibm_zos_cics.csd:
      state: changed
      input_location: INLINE
      input_content: |
        ADD GROUP(DFHTERMC) LIST(DFHLIST1)

        LIST LIST(DFHLIST1) OBJECTS

  - name: Provision a global catalog
    ibm.ibm_zos_cics.global_catalog:
      state: initial      You, 11 hours ago • additiona
```

region_jcl

The region_jcl module generates JCL to start a basic CICS region

Templates are useful here too, to describe naming conventions for region data sets, CICS install data sets, and le data sets

Note that cics_data_sets and le_data_sets use `<< lib_name >>` as the magic handle instead

```yaml
---
- name: "Use a template to describe the region data set naming conventions"

  gather_facts: false
  hosts: all
  environment: "{{ environment_vars }}"

  tasks:
    - name: "Generate CICS region JCL"
      ibm.ibm_zos_cics.region_jcl:
        state: initial
        applid: IYK2ZOES
        cics_data_sets:
          template: "CTS610.CICS740.<< lib_name>>"
          sdfhlic: "CTS610.CICS740.LIC.SDFHLIC"
        le_data_sets:
          template: "CEE.<< lib_name >>"      You, 11 hours ago • additional
        region_data_sets:
          template: "STEWF.RGNS.STARTC.<< data_set_name >>"
        job_parameters:
          region: 0M
        sit_parameters:
          start: INITIAL
          sit: 6$
          aicons: AUTO
          cicssvc: 217
          edsalim: 500M
          grplist: (DFHLIST*)
          gmtext: Region provisioned with Ansible
          srbsvc: 218
```

region_jcl

The resulting JCL has the templates resolved and applied for each data set name.

See the DD statements for STEPLIB, RPL and DFH* to see where the templates got applied

```
//IYK2ZOES JOB REGION=0M
//         EXEC PGM=DFHSIP,PARM=SI
//STEPLIB  DD DSN=CTS610.CICS740.SDFHAUTH,DISP=SHR
//         DD DSN=CTS610.CICS740.LIC.SDFHLIC,DISP=SHR
//         DD DSN=CEE.SCEERUN,DISP=SHR
//         DD DSN=CEE.SCEERUN2,DISP=SHR
//DFHRPL   DD DSN=CTS610.CICS740.SDFHLOAD,DISP=SHR
//         DD DSN=CEE.SCEECICS,DISP=SHR
//         DD DSN=CEE.SCEERUN,DISP=SHR
//         DD DSN=CEE.SCEERUN2,DISP=SHR
//DFHGCD   DD DSN=STEWF.RGNS.STARTC.DFHGCD,DISP=SHR
//DFHLCD   DD DSN=STEWF.RGNS.STARTC.DFHLCD,DISP=SHR
//DFHINTRA DD DSN=STEWF.RGNS.STARTC.DFHINTRA,DISP=SHR
//DFHLRQ   DD DSN=STEWF.RGNS.STARTC.DFHLRQ,DISP=SHR
//DFHTEMP  DD DSN=STEWF.RGNS.STARTC.DFHTEMP,DISP=SHR
//DFHAUXT  DD DSN=STEWF.RGNS.STARTC.DFHAUXT,DISP=SHR
//DFHBUXT  DD DSN=STEWF.RGNS.STARTC.DFHBUXT,DISP=SHR
//DFHDMPA  DD DSN=STEWF.RGNS.STARTC.DFHDMPA,DISP=SHR
//DFHDMPB  DD DSN=STEWF.RGNS.STARTC.DFHDMPB,DISP=SHR
//DFHCSD   DD DSN=STEWF.RGNS.STARTC.DFHCSD,DISP=SHR
//CEEMSG   DD SYSOUT=*
//CEEOUT   DD SYSOUT=*
//MSGUSR   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSABEND DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//DFHCXRF  DD SYSOUT=*
//LOGUSR   DD SYSOUT=*
//SYSIN    DD *
START=INITIAL
SIT=6$
```

CICS region provisioning sample

There’s a new sample in the shared samples repository for provisioning a basic CICS region

You can run full_provision.yml to allocate and initialize all the data sets for a basic CICS region, creates JCL, and submits it to start the region

The sample is expected to be modified to put something useful into the resulting region!

It makes heavy use of templating and module_defaults to be very concise!

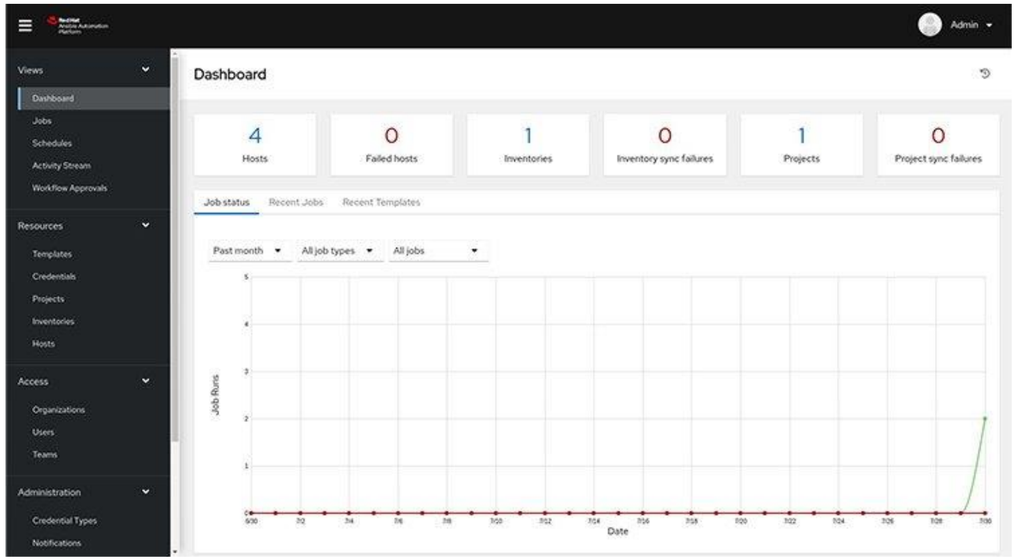
```
module_defaults:
  group/ibm.ibm_zos_cics.region:
    state: initial
    cics_data_sets:
      template: "CTS610.CICS740.<< lib_name >>"
      sdfhlic: "CTS610.CICS740.LIC.SDFHLIC"
    region_data_sets:
      template: "{{ ansible_user }}.REGIONS.{{ applid }}.<< data_set_name >>"
    le_data_sets:
      template: "CEE.<< lib_name >>"

tasks:
  - name: Create the auxiliary temporary storage data set (DFHTEMP)
    ibm.ibm_zos_cics.aux_temp_storage:

  - name: Create the auxiliary trace data set (DFHAUXT)
    ibm.ibm_zos_cics.aux_trace:

  - name: Create the second auxiliary trace data set (DFHBUXT)
    ibm.ibm_zos_cics.aux_trace:
      destination: B

  - name: Create the transaction dump data set (DFHDMPA)
    ibm.ibm_zos_cics.transaction_dump:

  - name: Create the second transaction dump data set (DFHDMPB)
    ibm.ibm_zos_cics.transaction_dump:
      destination: B

  - name: Create the CSD data set (DFHCSD)
    ibm.ibm_zos_cics.csd:
```

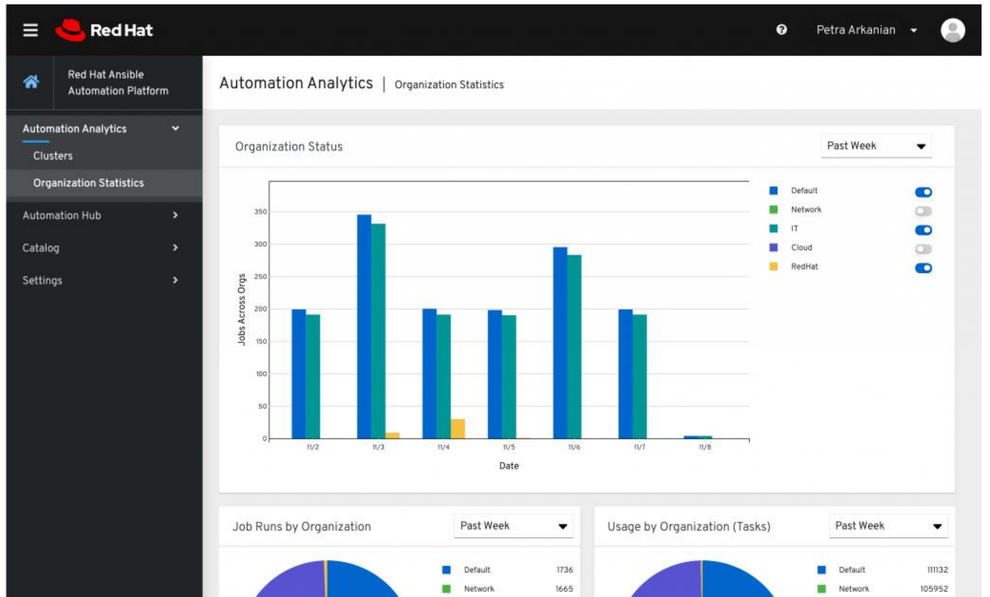
Automation as a service

By using Ansible Automation Platform across the enterprise you can take a higher-level view, without needing to know the specifics of exactly how individual playbooks are put together.

Users can use Automation Controller to run playbooks without having to install Ansible themselves, and run them with functional credentials.

They can also schedule automation to run at specific times, such as a monthly audit or a dashboard refresh every minute.





Scenario

A CICS application developer has created a new brand new COBOL application and wants to test it out

They don’t have the knowledge or skills to create a CICS environment but their CICS System programmer does.

Using a combination of Ansible, the CICS Ansible collection and Ansible Automation platform, can their system programmer can provide them with a better experience?

Getting started

To get going, go to the CICS collection on Ansible Galaxy, and follow the instructions to install Ansible and the collection.

Separately, ensure you have enabled CMCI on your target CICS regions or CICSplexes.

Use our ‘reporting’ sample to run your first playbook with the CICS collection.

Please submit your session feedback!

- All done via the Whova App
- QR Code to the right to download the Whova App

This session is EK