S503 – Ansible on Z 101

Drew Hughes
Software Developer – CICS Modernization
Andrew.Hughes1@ibm.com

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

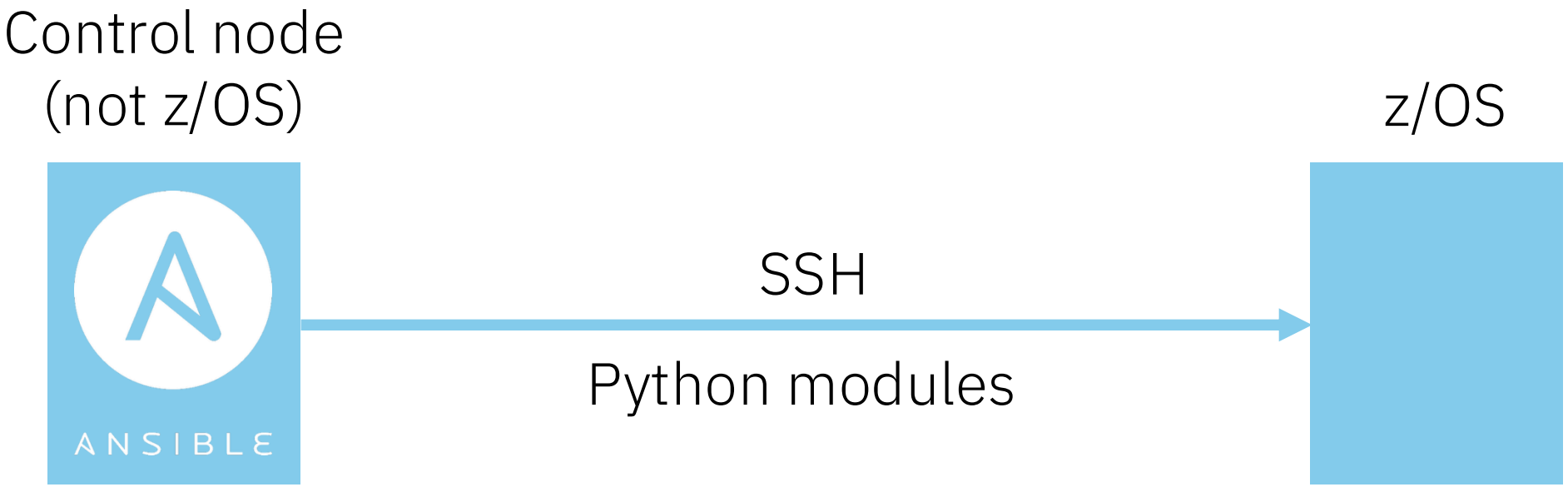
- System provisioning
- Installing applications
- Managing users
- Updating certificates
- Continuous delivery
- Configuration management

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

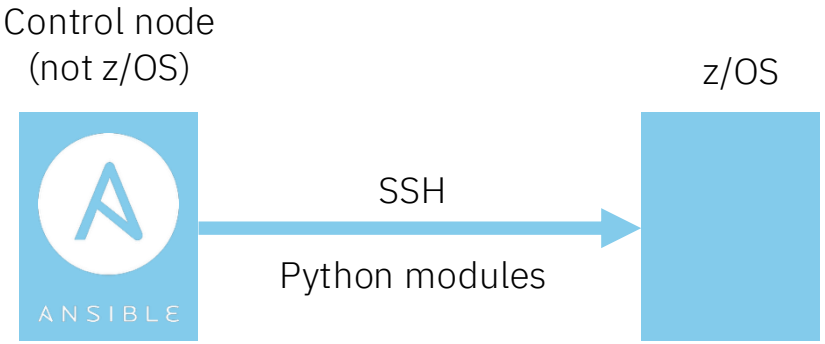


Ansible Glossary

- **Ansible** –The runtime that executes your automation. A Python application.
- **Ansible Playbook** - The ‘script’ that implements your automation. A dialect of YAML
- **Ansible Collection** – An extension to Ansible, installed using the ansible-galaxy CLI tool
- **Ansible Galaxy** – An online repository of 3000+ community developed Ansible extensions
- **Ansible Automation Platform** – An ecosystem of technologies to support your Ansible Automation strategy (a Red Hat offering)

Ansible the Runtime

```
ansible
```



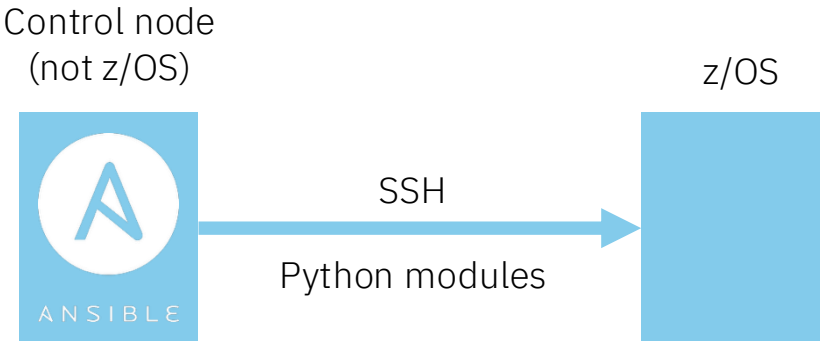
Ansible is a Python application which runs on the control node, and orchestrates automated processes on the managed nodes

Going to demonstrate how to set up Ansible for developing automation for z/OS, with a “Hello, world!” example

It’s one-time set-up (for everyone)

There are better ways of making your automation available to everyone (which we’ll come on to later)

Windows

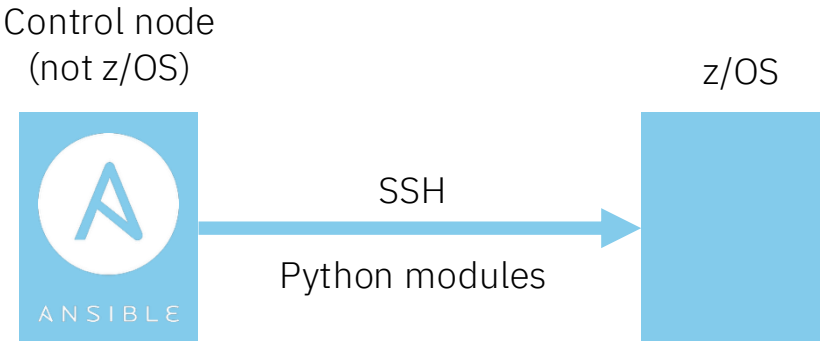


Ansible is native to Unix-like operating systems

This means Ansible doesn’t run natively in Windows

If you’re using Windows and want to do this, you’ll need to use WSL

Python



Installation instructions for Python will vary depending on which platform you're using

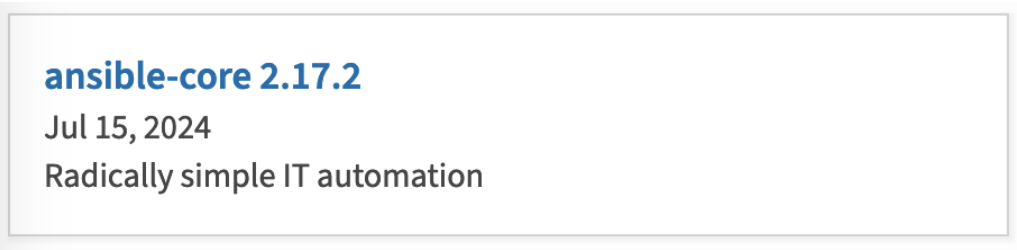
I prefer to use pyenv to manage different versions of the Python runtime

Latest version of Ansible requires >=Python 3.10

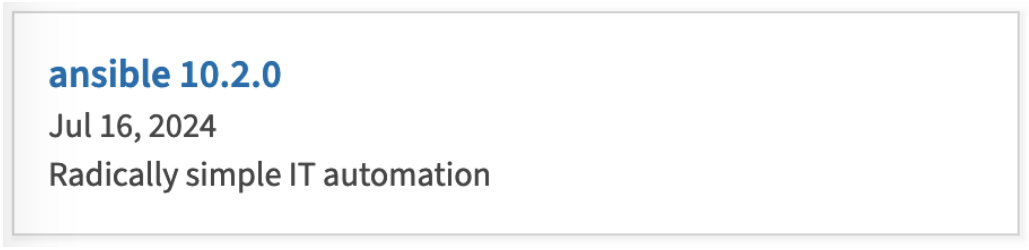
Installing Ansible

Ansible is installed from the online Python package repository PyPI (Python Package Index)

When you install Python, it comes with a CLI tool for installing packages, `pip`



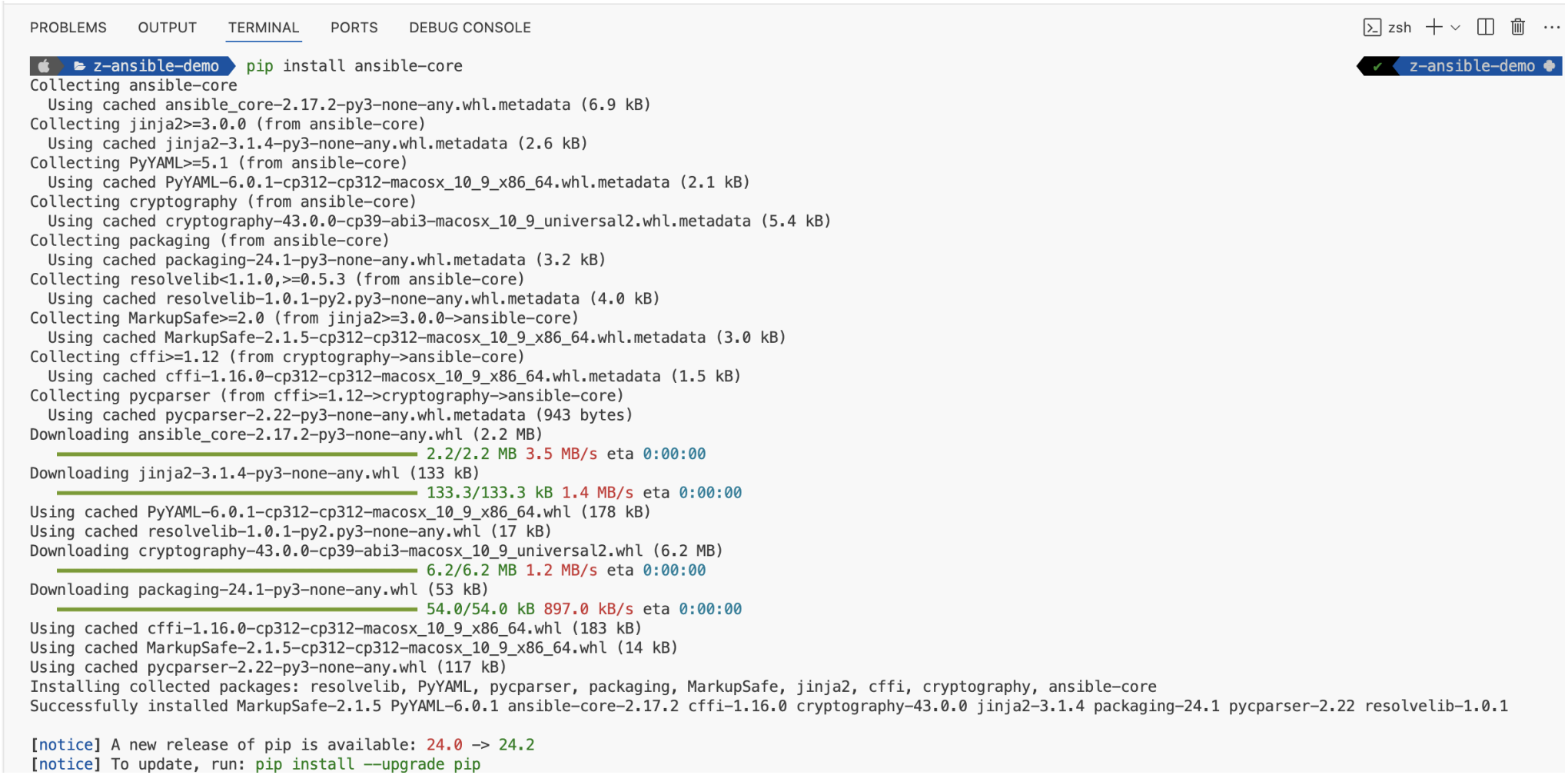
ansible runtime



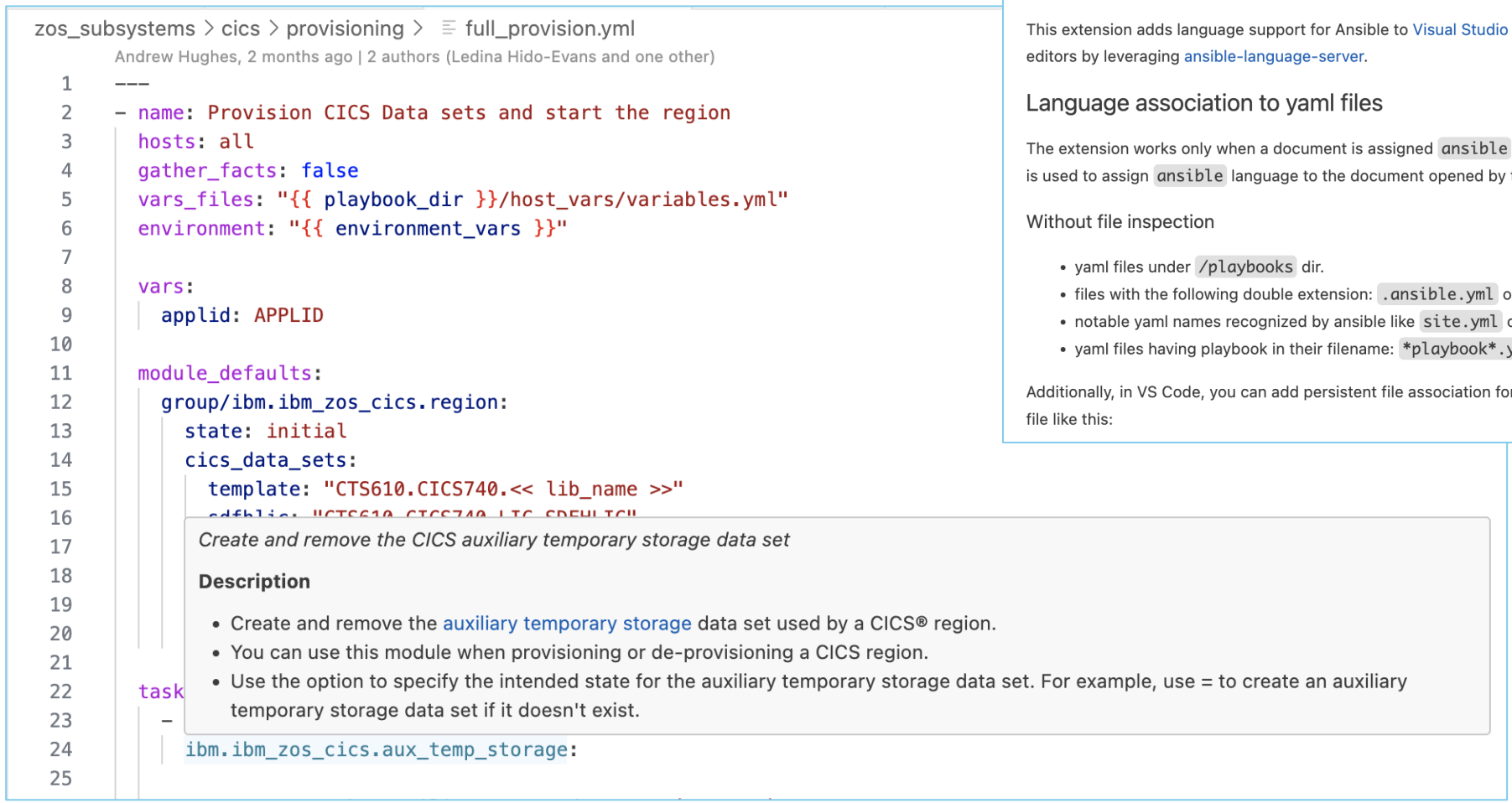
ansible runtime and ~85 community collections

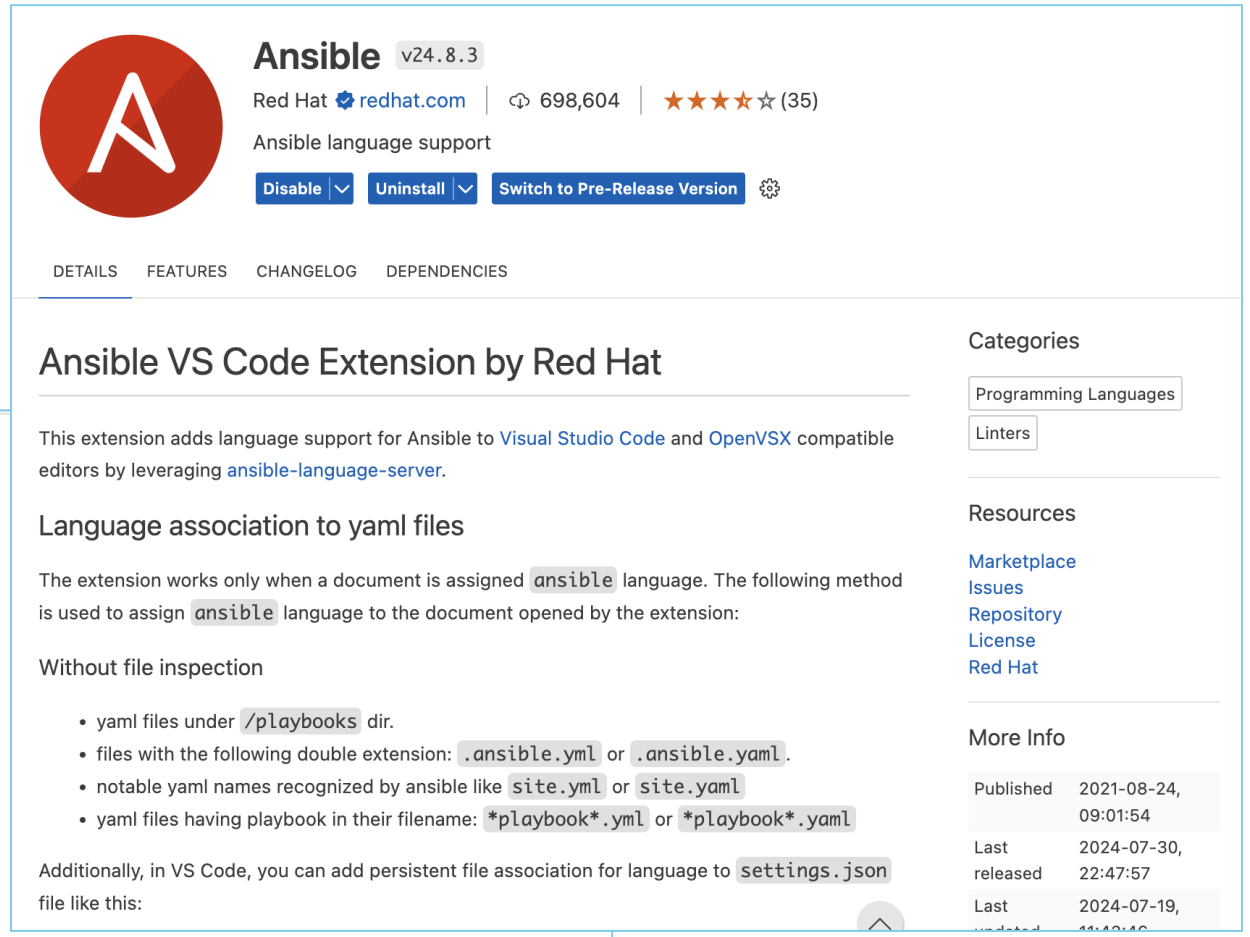
```
pip install ansible-core
```

Installing Ansible



VS Code


zos_subsystems > cics > provisioning > full_provision.yml
Andrew Hughes, 2 months ago | 2 authors (Ledina Hido-Evans and one other)
1 ---
2 - name: Provision CICS Data sets and start the region
3 hosts: all
4 gather_facts: false
5 vars_files: "{{ playbook_dir }}/host_vars/variables.yml"
6 environment: "{{ environment_vars }}"
7
8 vars:
9 applid: APPLID
10
11 module_defaults:
12 group/ibm.ibm_zos_cics.region:
13 state: initial
14 cics_data_sets:
15 template: "CTS610.CICS740.<< lib_name >>"
16
17
18
19
20
21
22 task
23 -
24 ibm.ibm_zos_cics.aux_temp_storage:
25
Create and remove the CICS auxiliary temporary storage data set
Description
Create and remove the auxiliary temporary storage data set used by a CICS® region.
You can use this module when provisioning or de-provisioning a CICS region.
Use the option to specify the intended state for the auxiliary temporary storage data set. For example, use = to create an auxiliary temporary storage data set if it doesn't exist.


Ansible v24.8.3
Red Hat redhat.com 698,604 (35)
Ansible language support
Disable Uninstall Switch to Pre-Release Version
DETAILS FEATURES CHANGELOG DEPENDENCIES
Ansible VS Code Extension by Red Hat
This extension adds language support for Ansible to Visual Studio Code and OpenVSX compatible editors by leveraging ansible-language-server.
Language association to yaml files
The extension works only when a document is assigned ansible language. The following method is used to assign ansible language to the document opened by the extension:
Without file inspection
yaml files under /playbooks dir.
files with the following double extension: .ansible.yml or .ansible.yaml.
notable yaml names recognized by ansible like site.yml or site.yaml
yaml files having playbook in their filename: *playbook*.yml or *playbook*.yaml
Additionally, in VS Code, you can add persistent file association for language to settings.json file like this:
Categories
Programming Languages
Linters
Resources
Marketplace
Issues
Repository
License
Red Hat
More Info
Published 2021-08-24, 09:01:54
Last released 2024-07-30, 22:47:57
Last 2024-07-19,

Playbooks

Ansible playbooks are YAML files

Each playbook:

- Selects a set of hosts from your inventory (coming up)
- Can set some variables
- Runs a list of tasks against those target systems

```yaml
---
- name: "Say Hello world"

  gather_facts: false
  hosts: all

  environment: "{{ z_environment_vars }}"

  tasks:
    - name: "Say hello world"
      ansible.builtin.command:
        cmd: echo "Hello, world!"
```

Running our first playbook

Ansible comes with a set of CLI tools for running your automation

We're going to use the `ansible-playbook` command to execute our new playbook

```
ansible-playbook –i
<inventory> <playbook>
```

Running First Playbook

```
 z-ansible-demo  ansible-playbook -i snippets/inventory1.yaml snippets/first-playbook.yaml -vv
zsh: /usr/local/bin/ansible-playbook: bad interpreter: /usr/local/opt/python@3.9/bin/python3.9: no such file or directory
ansible-playbook [core 2.17.2]
  config file = /Users/stewf/.ansible.cfg
  configured module search path = ['/Users/stewf/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /Users/stewf/repos/z-ansible-demo/venv/lib/python3.12/site-packages/ansible
  ansible collection location = /Users/stewf/.ansible/collections:/usr/share/ansible/collections
  executable location = /Users/stewf/repos/z-ansible-demo/venv/bin/ansible-playbook
  python version = 3.12.2 (main, Apr 30 2024, 05:51:17) [Clang 15.0.0 (clang-1500.3.9.4)] (/Users/stewf/repos/z-ansible-demo/venv/bin/python)
  jinja version = 3.1.4
  libyaml = True
Using /Users/stewf/.ansible.cfg as config file
Skipping callback 'default', as we already have a stdout callback.
Skipping callback 'minimal', as we already have a stdout callback.
Skipping callback 'oneline', as we already have a stdout callback.

PLAYBOOK: first-playbook.yaml **************************************************************************************************************************
1 plays in snippets/first-playbook.yaml

PLAY [Say Hello world] *********************************************************************************************************************************

TASK [Say hello world] *********************************************************************************************************************************
task path: /Users/stewf/repos/z-ansible-demo/snippets/first-playbook.yaml:8
changed: [mv28] => changed=true
  cmd:
  - echo
  - Hello, world!
  delta: '0:00:00.020322'
  end: '2024-08-05 11:26:01.566945'
  msg: ''
  rc: 0
  start: '2024-08-05 11:26:01.546623'
  stderr: ''
  stderr_lines: <omitted>
  stdout: Hello, world!
  stdout_lines: <omitted>

PLAY RECAP *********************************************************************************************************************************************
mv28                       : ok=1    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Inventory

How to tell Ansible about the remote systems you're targeting

You can also set per-host variables in your inventory

Some variables are built-in (like ansible_python_interpreter)

We'll also set some custom variables

```yaml
all:
  hosts:
    mv28:
      ansible_host: winmvs28.hursley.ibm.com
      ansible_user: stewf
      ansible_port: 22
      ansible_python_interpreter: /python/v3114/pyz/bin/python

      z_environment_vars:
        _BPXK_AUTOCVT: "ON"
        _CEE_RUNOPTS: "FILETAG(AUTOCVT,AUTOTAG) POSIX(ON)"
        _TAG_REDIR_ERR: "txt"
        _TAG_REDIR_IN: "txt"
        _TAG_REDIR_OUT: "txt"
        LANG: "C"
        PYTHONSTDINENCODING: "cp1047"
```

A note on code pages

Ansible core isn't "Z" aware so needs a bit of instruction to accommodate EBCDIC code pages

We'll set up a variable in our inventory which contains the magic environment variables we need

Note: This doesn't actually set the environment variables, we have to do that in our playbooks!

```
all:
  hosts:
    mv28:
      ansible_host: winmvs28.hursley.ibm.com
      ansible_user: stewf
      ansible_port: 22
      ansible_python_interpreter: /python/v3114/pyz/bin/python

      z_environment_vars:
        _BPXK_AUTOCVT: "ON"
        _CEE_RUNOPTS: "FILETAG(AUTOCVT,AUTOTAG) POSIX(ON)"
        _TAG_REDIR_ERR: "txt"
        _TAG_REDIR_IN: "txt"
        _TAG_REDIR_OUT: "txt"
        LANG: "C"
```

SSH

Last thing to do before we can run playbooks is set up authentication

Ansible does support password authentication, but it’s HIGHLY recommended to set up passwordless cryptographic key based authentication

Generate a key locally

```
ssh-keygen -t rsa -f ~/.ssh/stewf-zos -C
stewf-zos -b 2048
```

Copy to z/OS

```
ssh-copy-id -f -i stewf-zos
winmvs2c.hursley.ibm.com
```

~/.ssh/authorized_keys

`ssh-copy-id` copies your public key to a file in your z/OS user directory

~/.ssh/authorized_keys

```
~:>cat ~/.ssh/authorized_keys
ssh-rsa <text-encoded-key> stewf-zos
```

Holders of the private keys associated with these public keys will be able to authenticate as your user, so make sure you know what's in there

There are stringent permission requirements on the ~/.ssh directory to make sure other users on the shared system can modify this file

z/OS functionality

z/OS extensions for Ansible are distributed on Ansible Galaxy (galaxy. ansible.com)

They provide additional Ansible tasks (amongst other things) which extend Ansible to be able to manage z/OS resources

ibm_zos_core provides foundational capabilities e.g. submit a job, allocate data sets etc

Name	Type	Description
zos_data_set	module	Manage data sets
zos_ping	module	Ping z/OS and check dependencies.
zos_encode	module	Perform encoding operations.
zos_mount	module	Mount a z/OS file system.
zos_job_output	module	Display job output
zos_operator_action_query	module	Display messages requiring action
zos_find	module	Find matching data sets
zos_mvs_raw	module	Run a z/OS program.
zos_backup_restore	module	Backup and restore data sets and volumes
zos_unarchive	module	Unarchive files and data sets in z/OS.
zos_lineinfile	module	Manage textual data on z/OS
zos_copy	module	Copy data to z/OS
zos_volume_init	module	Initialize volumes or minidisks.
zos_apf	module	Add or remove libraries to Authorized Program Facility (APF)

Red Hat® Ansible® Certified Content for IBM Z®:

Available on Ansible Galaxy here and on Automation Hub			
IBM z/OS® Core		ibm_zos_core Provided by ibm 23 Modules 0 Roles 26 Plugins 0 Dependencies mvs z data_set 9 more	**401,000** Downloads
IBM z/OS® CICS®		ibm_zos_cics Provided by ibm 15 Modules 0 Roles 35 Plugins 0 Dependencies z zos cics 4 more	**23,000** Downloads
IBM z/OS® IMS™		ibm_zos_ims Provided by ibm 8 Modules 0 Roles 9 Plugins 1 Dependency mvs zos_ims z 11 more	**54,000** Downloads
IBM z/OS® Management Facility (z/OSMF)		ibm_zosmf Provided by ibm 3 Modules 13 Roles 4 Plugins 0 Dependencies mvs liberty_provisioning_template z	**13,500** Downloads
IBM Z® Hardware Management Console		ibm_zhmc Provided by ibm 28 Modules 0 Roles 1 Plugin 0 Dependencies z hmc ibm 2 more	**114,000** Downloads
IBM Z® System Automation		ibm_zos_sysauto Provided by ibm 0 Modules 2 Roles 0 Plugins 0 Dependencies z zos ibm 6 more	**4,100** Downloads

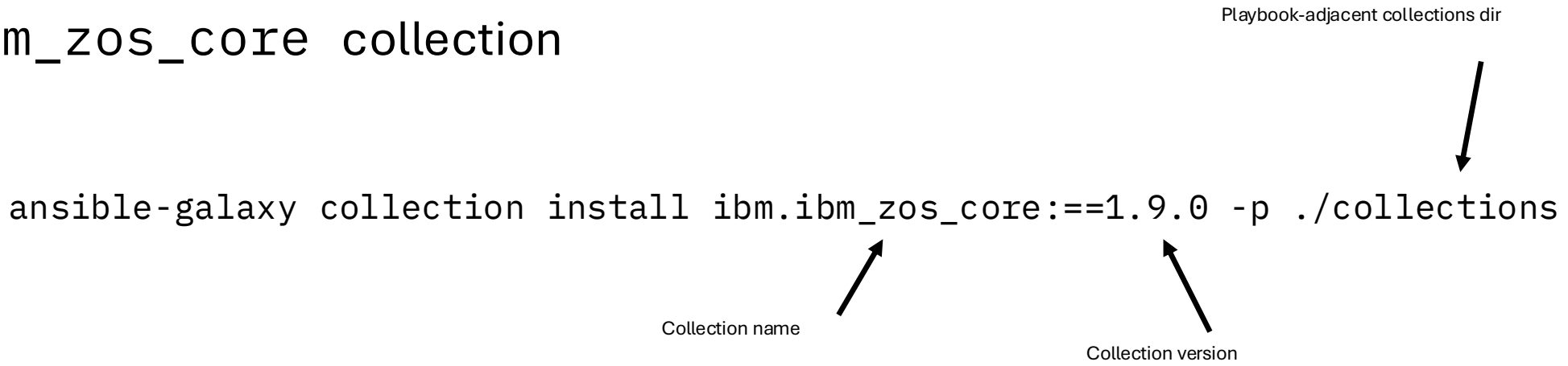
Collectively

- 83 Modules
- 15 Roles

Installing collections

Ansible comes with a separate CLI tool for managing installation of collections called `ansible-galaxy`

We'll use it to install the `ibm_zos_core` collection



IBM z/OS Open Automation Utilities

Many of the Ansible collections for z/OS rely on a supporting library called zoau

Ansible modules execute in Unix System Services, so they need some help to access MVS resources (like data sets, jobs, operator commands)

zoau provides that help

It's a collection of USS CLI utilities for working with MVS resources

And a set of Python APIs in order to support Ansible

Examples:

```
dls -l "STEWF.*"
```

```
/u/stewf:>dls -l "STEWF.*"
STEWF.ISP25121.SPFLOG1.LIST          PS  VA     125 P2P623
STEWF.ISP33598.SPFLOG1.LIST          PS  VA     125 P2P10E
STEWF.ISP33598.SPFTEMP0.CNTL         PS  FB      80 P2P100
STEWF.ISP58906.SPFLOG1.LIST          PS  VA     125 P2P0CF
```

```
jls '/EXPAUTO'
```

```
/u/stewf:>jls '/EXPAUTO'
EXPAUTO  EXPAUTOY JOB30874 CC        0000
EXPAUTO  EXPAUTOY JOB30864 CC        0000
EXPAUTO  EXPAUTOY JOB30744 CC        0000
EXPAUTO  EXPAUTOY JOB30743 CC        0000
EXPAUTO  EXPAUTOY JOB30742 CC        0000
EXPAUTO  EXPAUTOY JOB30714 CC        0000
EXPAUTO  EXPAUTOY JOB30701 CC        0000
```

Which ZOAU version to get?

There's quite a tight coupling between the version of the ibm_zos_core collection, and the version of ZOAU

The best reference for which versions are compatible is found on the integrated doc site here

Version	Control Node	Managed Node
1.14.x	• ansible-core >=2.15.x • Ansible >=8.0.x • AAP >=2.4	• z/OS V2R5 - V3Rx • z/OS shell • z/OS OpenSSH • IBM Open Enterprise SDK for Python • IBM Z Open Automation Utilities >=1.3.4, <1.4.0
1.13.x	• ansible-core >=2.15.x • Ansible >=8.0.x • AAP >=2.4	• z/OS V2R5 - V3Rx • z/OS shell • z/OS OpenSSH • IBM Open Enterprise SDK for Python • IBM Z Open Automation Utilities >=1.3.3, <1.4.0
1.12.x	• ansible-core >=2.15.x • Ansible >=8.0.x • AAP >=2.4	• z/OS V2R5 - V3Rx • z/OS shell • z/OS OpenSSH • IBM Open Enterprise SDK for Python • IBM Z Open Automation Utilities >=1.3.2, <1.4.0
1.11.x	• ansible-core >=2.15.x • Ansible >=8.0.x • AAP >=2.4	• z/OS V2R4 - V2Rx • z/OS shell • z/OS OpenSSH • IBM Open Enterprise SDK for Python • IBM Z Open Automation Utilities >=1.3.1, <1.4.0

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

z/OS Scenarios - DFHCSDUP

Let’s try and automate a simple CICS scenario with the ibm_zos_core collection

We’ll use the zos_job_submit module, to run the DFHCSDUP program to list the contents of a CSD group

We’ll write a simple Jinja template to generate the right JCL

```yaml
---
- name: First z/OS playbook

  gather_facts: false
  hosts: all

  environment: "{{ z_environment_vars }}"

  vars:
    csdup_group: DFH$EXBS
    dfhcsd: STEWF.RGNS.IYK2ZOE3.DFHCSD

  tasks:
    - name: "Run the job template"
      ibm.ibm_zos_core.zos_job_submit:
        location: LOCAL
        use_template: true
        src: csdup.j2
```

z/OS Scenarios

PROBLEMS OUTPUT TERMINAL PORTS DEBUG CONSOLE

```
- '      '
- ' SUMMARY OF GROUPS                                                    UTILITY COMMAND:    LIST GROUP(DFH$EXBS)    OBJECTS'
- '      '
- ' ************************************************************************************************************************'
- '      '
- ' GROUP NAME: DFH$EXBS    (IBM PROTECTED)'
- ' ----------'
- '      '
- ' FILES:              EXMPCAT   EXMPCONF'
- ' MAPSETS:            DFH0XS1   DFH0XS2   DFH0XS3'
- ' PROGRAMS:           DFH0XCFG  DFH0XCMN  DFH0XGUI  DFH0XODE  DFH0XSDS  DFH0XSOD  DFH0XSSM  DFH0XVDS  DFH0XWOD'
- '      '
- ' TRANSACTIONS:       ECFG      EGUI'
- 1CICS RDO OFF-LINE UTILITY PROGRAM DFHCSDUP RELEASE:0740 PTF:HCI7400 .            TIME 02:54 DATE 24.219      PAGE 0002
- '-     '
- ' ************************************************************************************************************************'
- '      '
- ' OBJECTS IN GROUPS                                                    UTILITY COMMAND:    LIST GROUP(DFH$EXBS)    OBJECTS'
- '      '
- ' ************************************************************************************************************************'
- '      '
- ' GROUP NAME: DFH$EXBS    (IBM PROTECTED)'
- ' ----------'
- '      '
- ' FILE(EXMPCAT)          GROUP(DFH$EXBS)                                                                    24.157 00:27'
- '                                DESCRIPTION(CICS Example Application - Catalog File)'
- '                        VSAM-PARAMETERS'
- '      '
- '                                DSNAME(HLQ.EXMPLAPP.EXMPCAT)'
- '                                RLSACCESS(NO)          LSRPOOLNUM(1)          READINTEG(UNCOMMITTED)'
- '                                DSNSHARING(ALLREQS)    STRINGS(1)             NSRGROUP()'
- '                        REMOTE-ATTRIBUTES'
- '                                REMOTESYSTEM()         REMOTENAME()'
- '                        REMOTE-AND-CFDATATABLE-PARAMETERS'
- '                                RECORDSIZE()           KEYLENGTH()'
```

z/OS Scenarios – DFHCSDUP via zos_mvs_raw

The zos_mvs_raw module gives us a different option for executing z/OS programs – one without going via JES

We can convert the JCL template from the previous example into Ansible configuration, cutting out the templating entirely

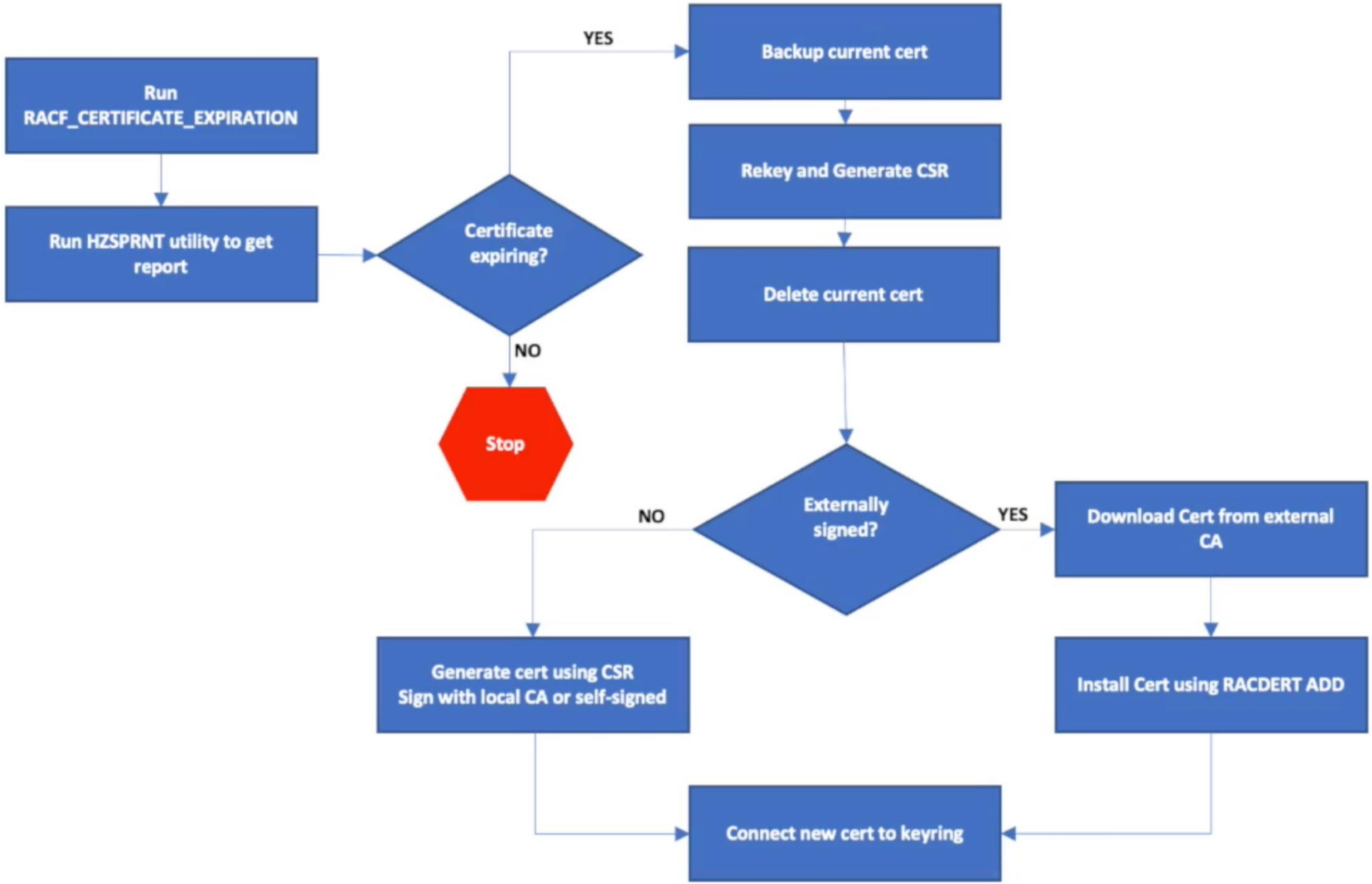
```yaml
---
- name: First z/OS playbook

  gather_facts: false
  hosts: all

  environment: "{{ z_environment_vars }}"

  vars:
    csdup_group: DFH$EXBS
    dfhcsd: STEWF.RGNS.IYK2ZOE3.DFHCSD

  tasks:
    - name: "Run the job template"
      ibm.ibm_zos_core.zos_mvs_raw:
        program_name: DFHCSDUP
        dds:
          - dd_data_set:
              dd_name: STEPLIB
              disposition: shr
              data_set_name: CTS610.CICS740.SDFHLOAD
          - dd_data_set:
              dd_name: DFHCSD
              disposition: shr
              data_set_name: "{{ dfhcsd }}"
          - dd_output:
              dd_name: SYSPRINT
              return_content:
                type: text
          - dd_input:
              dd_name: SYSIN
              content: " LIST GROUP({{ csdup_group }}) OBJECTS"
```

z/OS Scenarios – Certificate Renewal

The Ansible Z Sample Playbook GitHub repository is a shared samples repo for all the IBM Z collections for Ansible

There are documented examples there for working with e.g. ibm_zos_core, ibm_zos_cics, ibm_zos_ims, etc

Let’s take a look at the certificate renewal sample


Run RACF_CERTIFICATE_EXPIRATION
Run HZSPRNT utility to get report
Certificate expiring?
YES
NO
Stop
Backup current cert
Rekey and Generate CSR
Delete current cert
Externally signed?
NO
YES
Download Cert from external CA
Generate cert using CSR
Sign with local CA or self-signed
Install Cert using RACDERT ADD
Connect new cert to keyring

```
 SDSF OUTPUT DISPLAY RACF_CERTIFICATE_EXPIRATION   LINE 0        COLUMNS 02- 81
 COMMAND INPUT ===> _                                          SCROLL ===> PAGE
********************************* TOP OF DATA ***********************************
CHECK(IBMRACF,RACF_CERTIFICATE_EXPIRATION)
SYSPLEX:     XESDEV     SYSTEM: EC01129A
START TIME: 05/19/2022 07:17:50.173343
CHECK DATE: 20111010  CHECK SEVERITY: MEDIUM
CHECK PARM: days(366)

                  Certificates Expiring within 366 Days

S Cert Owner    Certificate Label                    End Date   Trust Rings
- ------------ ------------------------------------ ---------- ----- -----
  CERTAUTH     Verisign Class 1 Individual CA       2008-05-12 No        0
  CERTAUTH     Thawte Personal Basic CA             2020-12-31 No        0
  CERTAUTH     Thawte Personal Freemail CA          2020-12-31 No        0
  CERTAUTH     Thawte Personal Premium CA           2020-12-31 No        0
E CERTAUTH     zOSMFCA                              2023-05-18 Yes       1
E CERTAUTH     ANDY CA                              2023-05-20 Yes       1
  CERTAUTH     Thawte Premium Server CA             2020-12-31 No        0
  CERTAUTH     Thawte Server CA                     2020-12-31 No        0
E ID(IZUSVR)   DefaultzOSMFCert.IZUDFLT             2023-05-18 Yes       1
  CERTAUTH     ICP-Brasil CA v1                     2021-07-29 No        0
E SITE         ANDY SITE                            2022-06-16 Yes       1
  CERTAUTH     GTE CyberTrust Root CA               2006-02-23 No        0
E ID(OMVSADM)  ANDY SSL                             2022-06-16 Yes       1
  CERTAUTH     RSA Secure Server CA                 2010-01-07 No        0
  CERTAUTH     GeoTrust Global CA                   2022-05-21 No        0
  CERTAUTH     Entrust.net Secure Server CA         2006-01-01 No        0
  CERTAUTH     ICP-Brasil CA                        2011-11-30 No        0
  CERTAUTH     Verisign International Svr CA        2011-10-24 No        0
  CERTAUTH     Verisign Class 1 Primary CA          2020-01-07 No        0
  CERTAUTH     Integrion CA                         2017-05-20 No        0
  CERTAUTH     IBM World Registry CA                2017-05-20 No        0
```

z/OS Scenarios – Certificate Renewal

```yaml
tasks:
  - name: Run health check via operator command.
    tags: search_and_renew, search, run_hc
    ibm.ibm_zos_core.zos_operator:
      cmd: "F HZSPROC,RUN,CHECK=(IBMRACF,RACF_CERTIFICATE_EXPIRATION)"

  - name: Get sysname.
    tags: search_and_renew, search, search_hc
    ansible.builtin.command:
      cmd: sysvar SYSNAME
    changed_when: false
    register: sysname

  - name: Submit job to pull health check report.
    tags: search_and_renew, search, search_hc
    ibm.ibm_zos_core.zos_job_submit:
      src: "{{ playbook_dir }}/templates/HZPRINT.jcl.j2"
      location: LOCAL
      wait_time_s: 60
    register: hc_job_output
```

Certificate Management playbook

There’s a great video which explains the search_and_renew.yml playbook on mediacenter

The video also shows how to provide a piece of Ansible automation as-a-service with Ansible Automation Controller

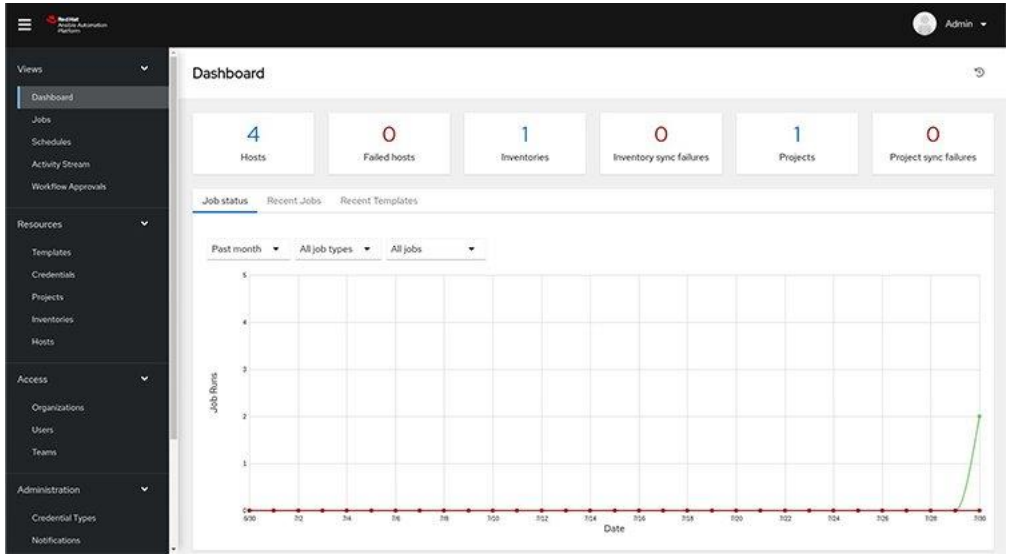
Check out the sample in the Z Ansible Playbook Samples repo

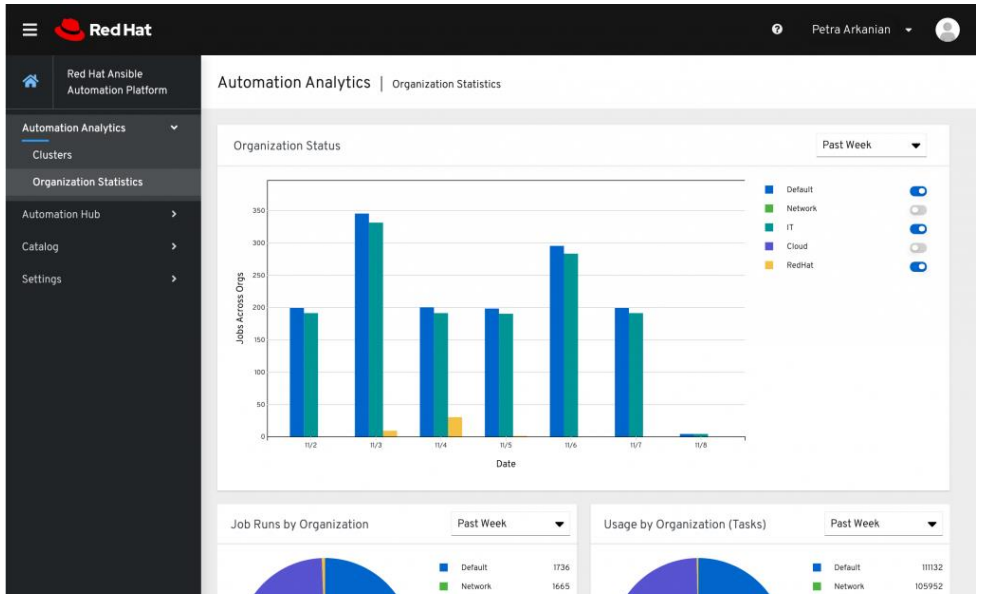
Automation as a service

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.

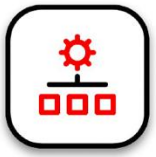
Users can use Automation Controller to run playbooks without having to install Ansible themselves, and run them with functional credentials.

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





One Subscription. One integrated platform.



Automation controller
Automation control plane

Automation execution environments
Scalable packaging and runtime execution plane

Automation mesh
Connectivity across diverse enterprise automation environments

NEW

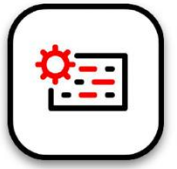
Event-Driven Ansible
Automatic response to environment changes based on environment intelligence

Ansible-builder
Ansible containerized execution environment builder

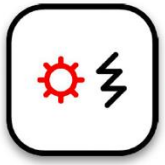
Automation analytics & Red Hat Insights
Visibility, predictive analytics, and more

Ansible Content Collections
100+ certified content collections

Automation hub
Hosted certified content repository.



Ansible-navigator
Execution environment orchestration tooling



Ansible Platform Operator
Package, deploy and manage this platform on Red Hat OpenShift

Microsoft VS code plugin
Write and manage Ansible code with Visual Studio

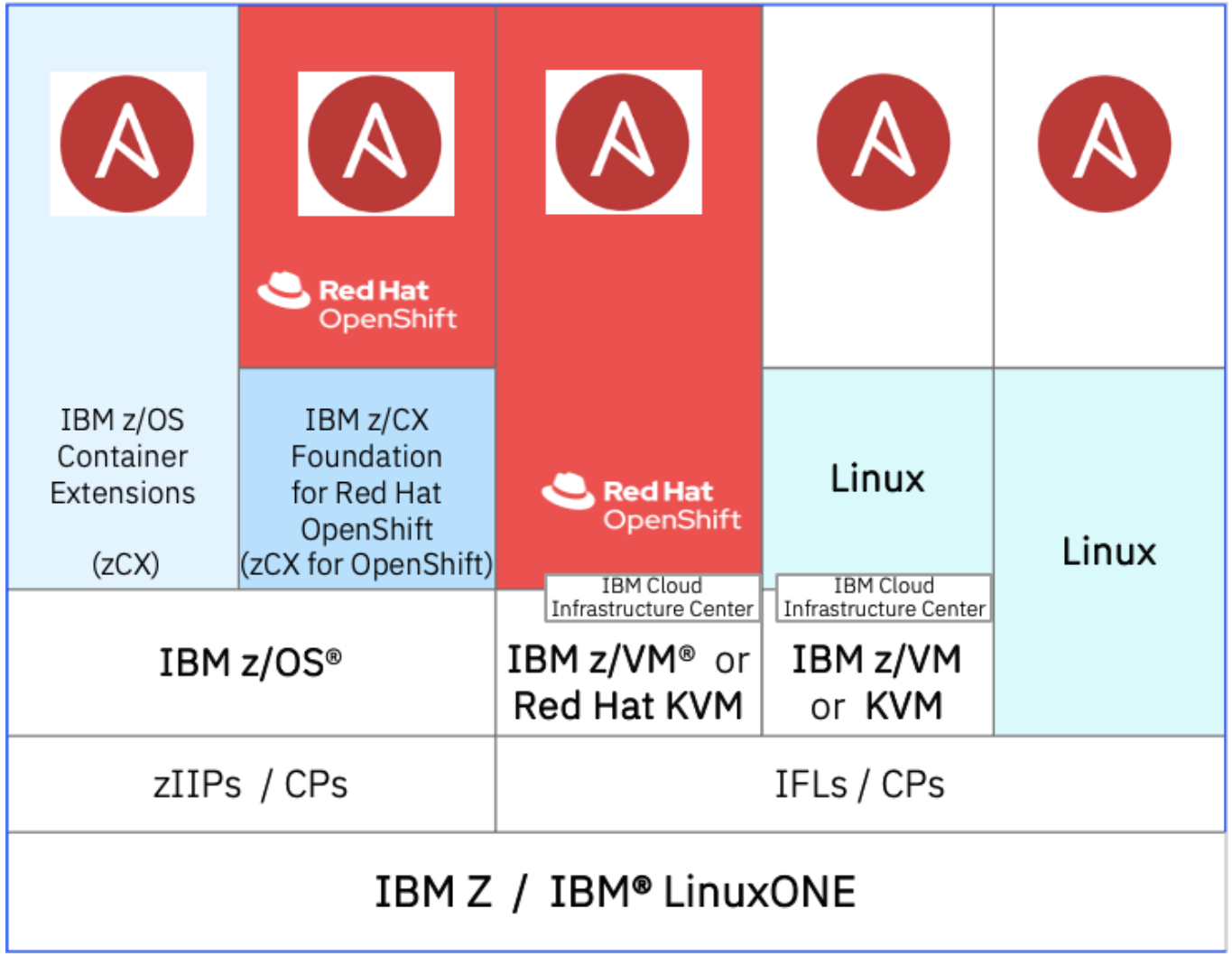
Red Hat
Ansible Automation
Platform

Red Hat
Ansible Automation
Platform

Red Hat Ansible Automation Platform (AAP)

Deployment options on IBM Z / IBM LinuxONE

- Linux (native LPAR, z/VM, KVM, and/or ICIC)
- Red Hat OpenShift Container Platform
- IBM zCX Foundation for Red Hat OpenShift
- IBM z/OS Container Extensions (zCX)




IBM