CICS DevOps with Ansible

Stewart Francis - stewartfrancis@uk.ibm.com

Acknowledgements

Agenda

Today we'll be looking at **Ansible** for z/OS and especially how that relates to CICS Transaction Server.

- What is Ansible and what support does it have on z/OS?
- Introduction to the CICS collection for Ansible

About Ansible — in general, and for z/OS

About 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?

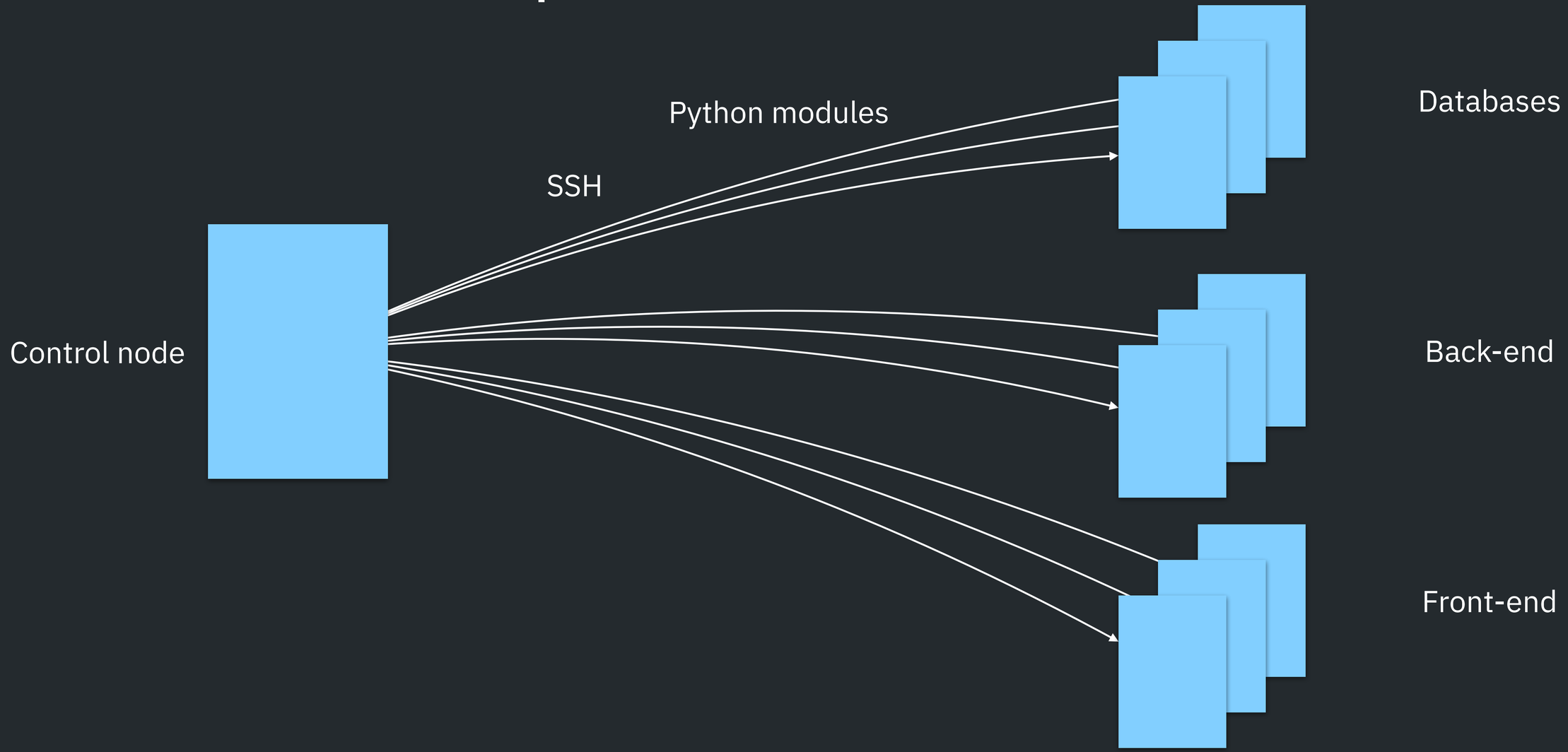
Ansible is extremely popular across many enterprises. For example, it’s now the top cloud configuration tool and is heavily used on-prem too.

Some of the reasons for that include:

- It normalizes tooling across a multitude of platforms
- It centralizes your enterprise automation strategy
- You can achieve configuration as code
- Over 3000 modules for all the things you might need it to do

If you’re not aware of it already in your enterprise, ask around!

The basic concepts of Ansible



The Ansible inventory

The inventory identifies the nodes that are managed, and categorises them into different groups.

Both nodes and groups can be assigned variables for use later during automation.

Inventory can be static (via a file) or dynamically provided.

```
back-end.example.com

[databases]
db1.example.com
db2.example.com
db3.example.com

[front-end]
fe1.example.com
fe2.example.com
fe3.example.com
```

Ansible playbooks

Playbooks bind hosts from the inventory to tasks

The hosts can be individual nodes, groups or everything

When a playbook is executed, it runs tasks against the hosts they’re bound to

```
---
- name: update web servers
 hosts: webservers
 remote_user: root

 tasks:
  - name: ensure apache is at the latest versi
    yum:
      name: httpd
      state: latest
  - name: write the apache config file
    template:
   src: /srv/httpd.j2
   dest: /etc/httpd.conf

- name: update db servers
 hosts: databases
  remote_user: root

  tasks:
  - name: ensure postgresql is at the latest v
    yum:
      name: postgresql
      state: latest
  - name: ensure that postgresql is started
```

Ansible tasks

Tasks are work that you configure to be run on the managed nodes.

Some are built into Ansible, but many are provided by Ansible plugins.

Tasks are configured using Ansible, with configuration that is relevant to that task.

```
- shell:
  chdir: somedir/
  cmd: ls -l | grep log
```

Running a playbook

To run Ansible, it needs to be installed on a control node.

The playbook can then be run from the command line.

The tasks etc — composed of Python modules — are shipped from the control node to the managed nodes.

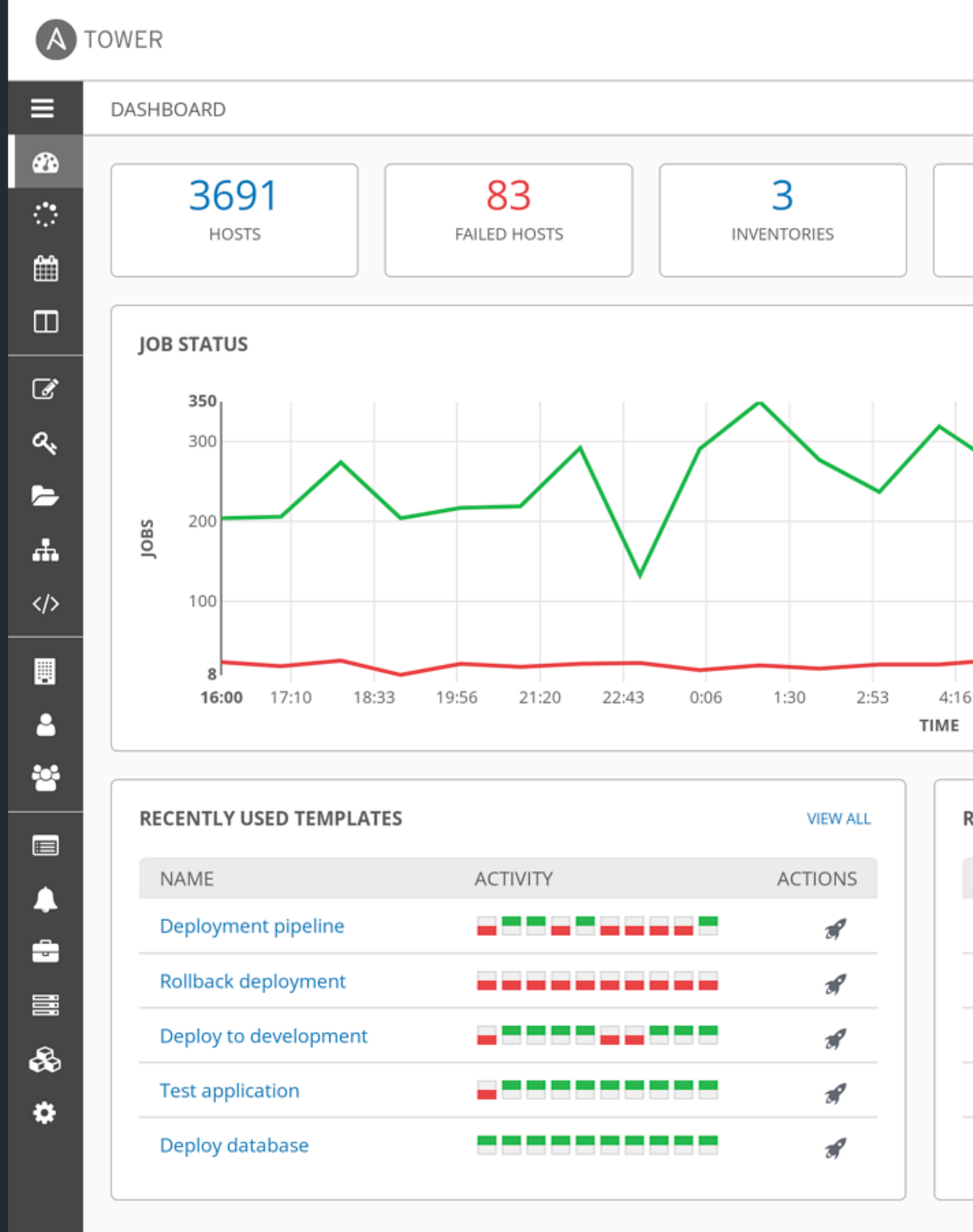
```
ansible-playbook
  -i databases,
  -e varname=varval
  my_playbook.yml
```

Automation as a service

By using Ansible Tower 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 Tower 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.



Why would you want to use Ansible on z/OS?

Using Ansible on z/OS allows you to centralise your automation skill set around a particular open source technology that gives you flexibility and power.

By sharing the same automation strategy as the rest of the enterprise you can unlock opportunities for collaboration and integration.

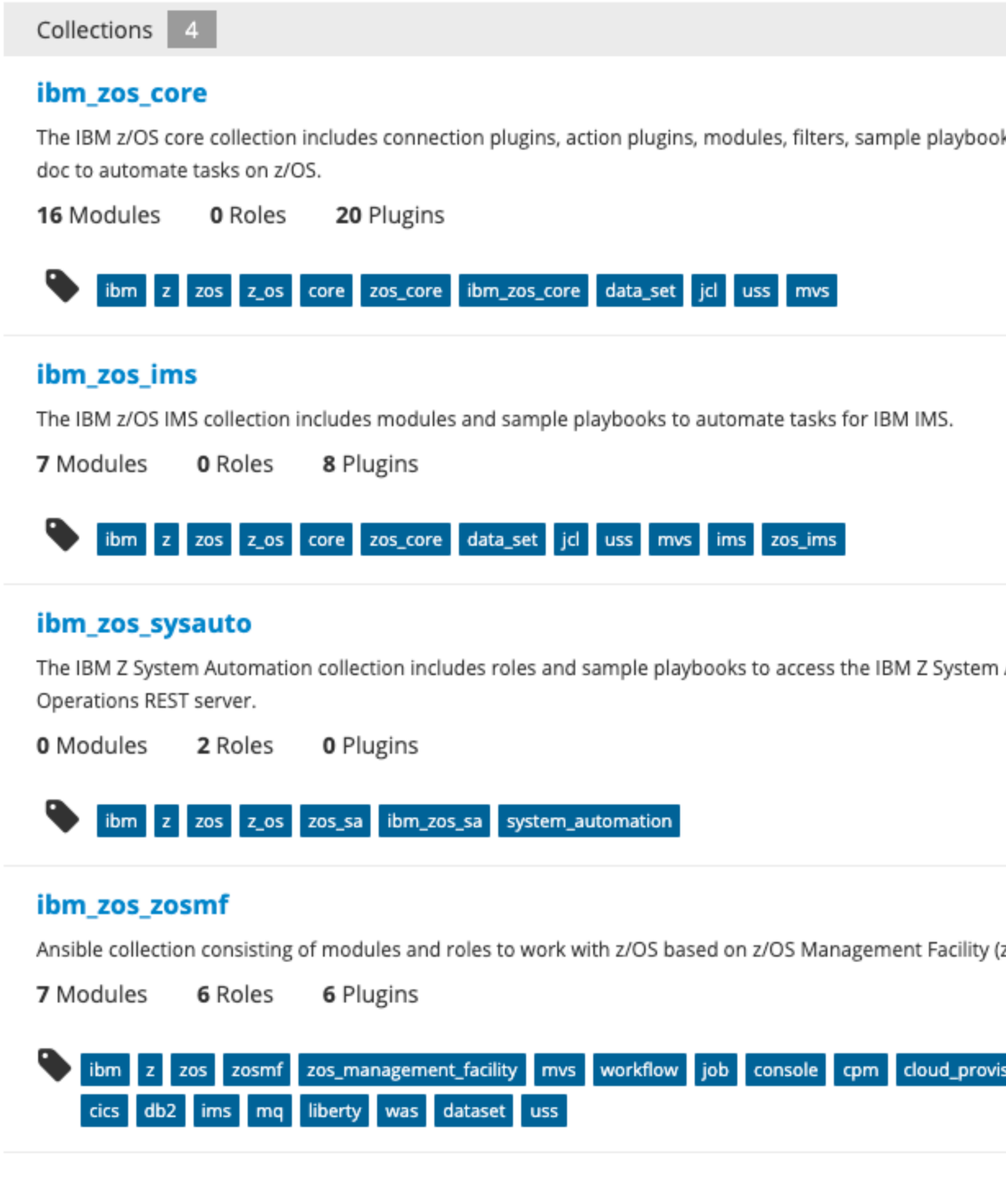
Ansible’s flexibility permits reuse of your existing automation — triggering System Automation (SA), z/OSMF workflows, JCL, … — or adaptation to specific Ansible tasks.

Ansible for z/OS

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 Hub 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.



The CICS TS collection for Ansible

We’re in the process of developing a CICS collection for Ansible — bringing 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 will be part of the (supported) Red Hat Ansible Certified Content for IBM Z.

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

Tasks in the CICS collection

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 1

A regular need as a CICS system programmer is to save away the details of which resources or definitions are available in a particular environment, perhaps as a CSV ready for import in a spreadsheet.

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

Scenario 2

Application deployment, and in particular the movement and NEWCOPYing of PROGRAM resources, is perhaps one of the most common forms of automation used for CICS.

How would a system programmer automate moving dataset members from the build PDSE to the production CICS DFHRPL or LIBRARY PDSE, and then NEWCOPY the relevant programs?

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.

Thank you! Any questions?