How to drive your webservices with Ansible

And Why! ...

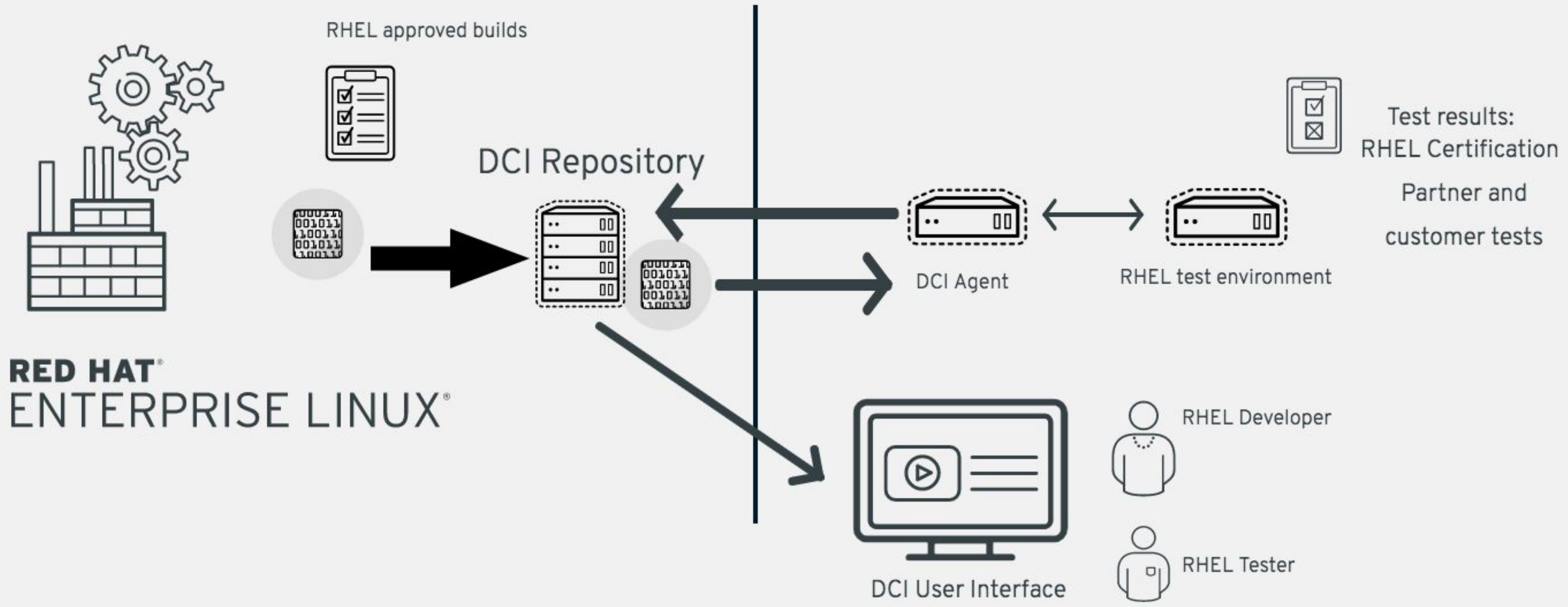
Goneri Le Bouder

Ansible Montréal - 2018/10

Some context

DISTRIBUTED CI

DCI FOR RHEL


RHEL approved builds
DCI Repository
Test results:
RHEL Certification
Partner and
customer tests
DCI Agent
RHEL test environment
RED HAT
ENTERPRISE LINUX
RHEL Developer
RHEL Tester
DCI User Interface

What we tried to resolve

- Need a way to interact with our resources
- Should be easily readable by a non-developer audience
- We don’t want to do some shell scripting on top of our CLI

Why Ansible?

Ansible was already popular in the team

- We use it to manage the production environment
- Well integrated in our CI/CD chain

Our users were already

- Familiar with it
- Or willing to learn

Lingua franca internally for the deployment of the product deployment

- Ceph-Ansible
- OpenShift-Ansible
- etc

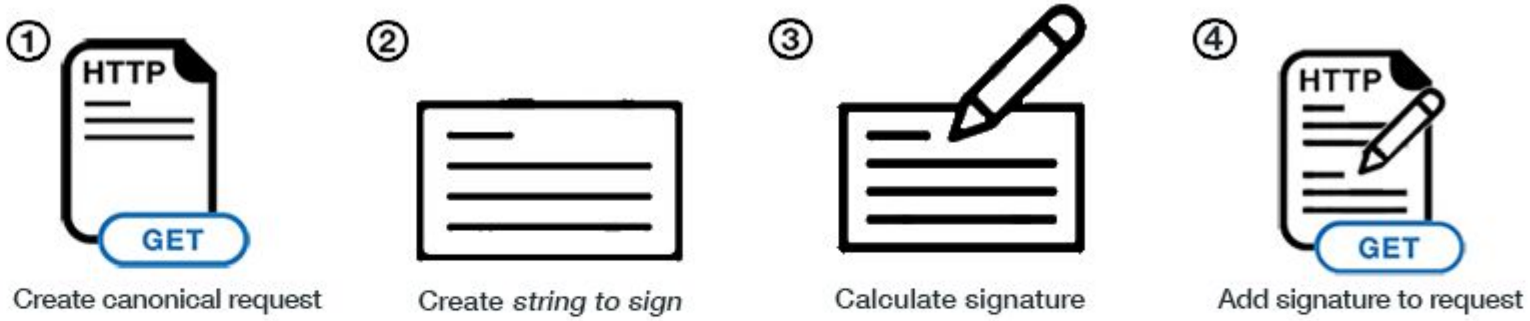
So we will prepare our own modules

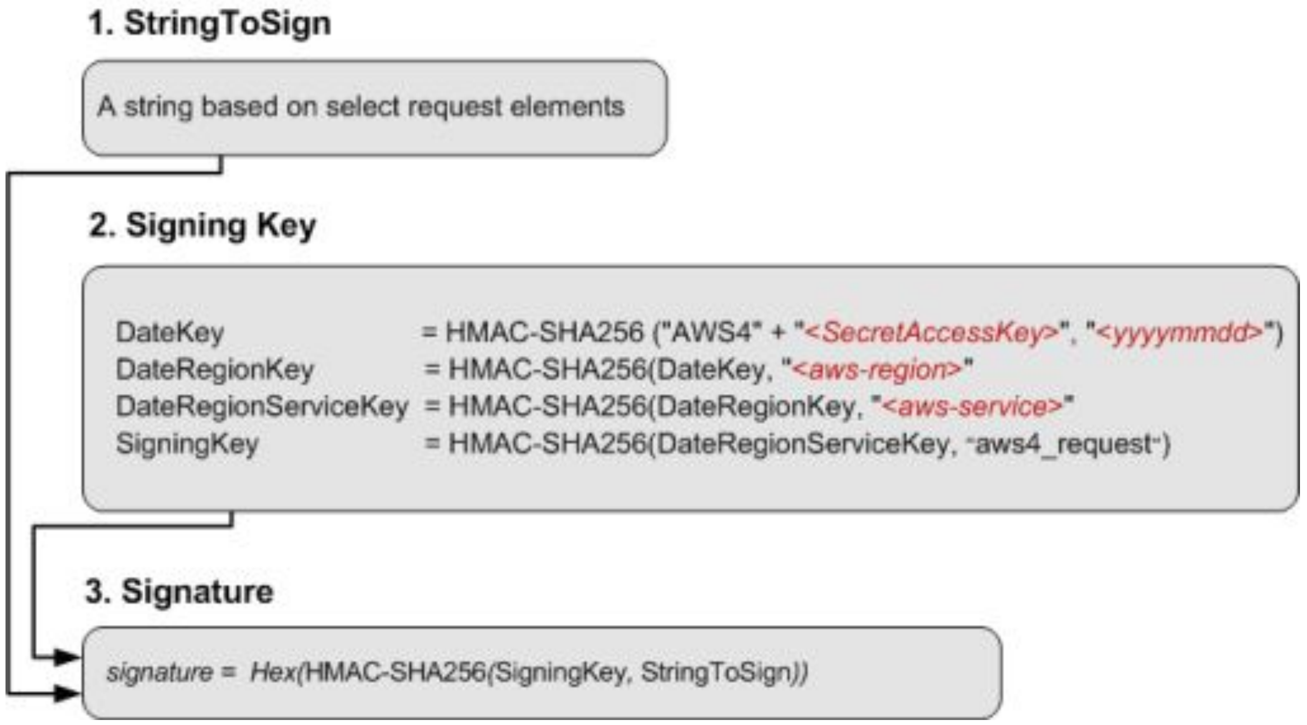
But! The uri module already does that?!

```
- name: Create a JIRA issue
  uri:
    url: https://your.jira.example.com/rest/api/2/issue/
    method: POST
    user: your_username
    password: your_pass
    body: "{{ lookup('file','issue.json') }}"
    force_basic_auth: yes
    status_code: 201
    body_format: json
```

uri was not an option (1/2)

- Authentication layer
    - We use AWS Signature Version 4


①
HTTP
GET
Create canonical request
②
Create *string to sign*
③
Calculate signature
④
HTTP
GET
Add signature to request



1. **StringToSign**

A string based on select request elements

2. **Signing Key**

```
DateKey                  = HMAC-SHA256 ("AWS4" + "<SecretAccessKey>", "<yyyymmdd>")
DateRegionKey            = HMAC-SHA256(DateKey, "<aws-region>"
DateRegionServiceKey     = HMAC-SHA256(DateRegionKey, "<aws-service>"
SigningKey               = HMAC-SHA256(DateRegionServiceKey, "aws4_request")
```

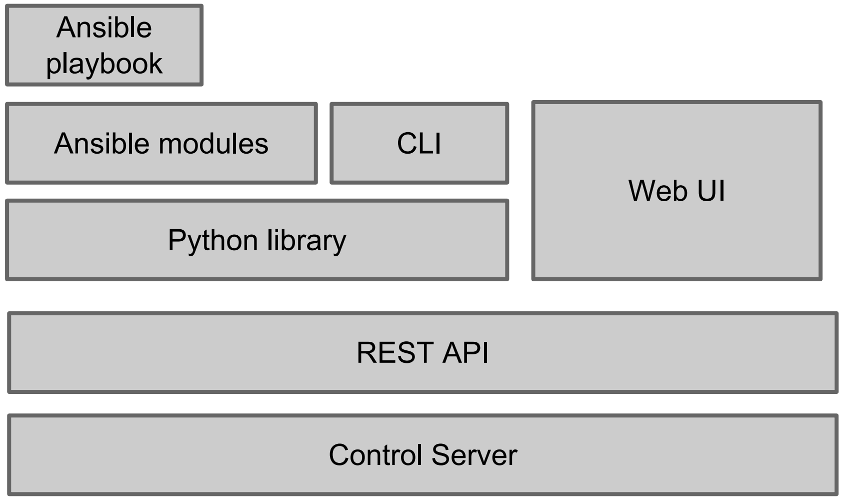
3. **Signature**

signature = *Hex(*HMAC-SHA256*(*SigningKey, StringToSign*))*

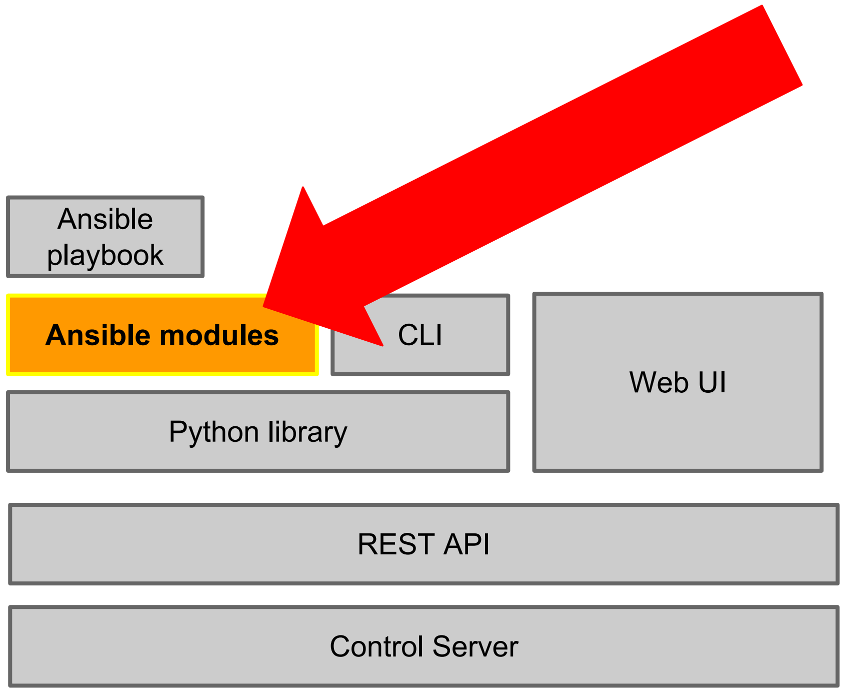
uri was not an option (2/2)

- Authentication layer
  - We use AWS Signature Version 4
- Imply boilerplate code
  - to handle errors
  - format some parameters
- ...

Our ~~final~~ current technical stack



Our ~~final~~ current technical stack



Our API

- Very generic REST API
- 10 =~ resources
- We use the standard REST verbs

Our API: list

```
GET http://srv/api/v1/roles
```

Our API: list

```
POST http://srv/api/v1/roles
Content-Type: application/json

{
    “Name“: “boby”
}
```

Our API: get

```
GET http://srv/api/v1/roles/$foo
```

Our API: delete

```
DELETE http://srv/api/v1/roles/$foo
```

From Ansible

You can adjust your ansible.cfg to include another module directory (library). e.g:

```
[defaults]
library          = /usr/share/dci/modules/
```

Python code sample

```
def main():
    resource_argument_spec = dict(
(blabla)
    )
    resource_argument_spec.update(authentication_argument_spec())
    module = AnsibleModule(
        argument_spec=resource_argument_spec,
        required_if=[['state', 'absent', ['id']]]
    )
    context = build_dci_context(module)
    action_name = get_standard_action(module.params)
    role = DciRole(module.params)
    action_func = getattr(role, 'do_%s' % action_name)
    http_response = run_action_func(action_func, context, module)
    result = parse_http_response(http_response, dci_role, context, module)
    module.exit_json(**result)
```

How to share code between modules?

If you have several modules like us, you may want to share some code between them. The module_utils directory can be handle:

```
[defaults]
library            = /usr/share/dci/modules/
module_utils       = /usr/share/dci/module_utils/
```

How to share code between modules?

In our case, we share a dci_common.py for:

- Error handling
- Boilerplate for the different actions (delete, list, get, update, etc)
- Authentication
- And argument parsing

Idempotence

Reentrancy is import (much like a regular playbook)

You should be able to rerun the same module with the same parameters.

Documentation

Ansible-doc will read your module documentation.

https://docs.ansible.com/ansible/2.7/dev_guide/developing_modules_documenting.html

Testing (1/2)

- Hard to do unit-testing
  - We actually gave up
- We redeploy an testing environment
  - Molecule is not an option AFAIK

- “Unit-testing” through a series of task/assert
  - more like integration testing with a limited scope
- Functional testing
  - A playbook to
  - Serie of playbook