WRITING ANSIBLE MODULES FOR FOREMAN AND KATELLO
$ WHOAMI
Evgeni Golov
Senior Software Engineer at Red Hat
ex-Consultant at Red Hat
Debian and Grml Developer
♥ FOSS ♥
♥ automation ♥
WTF?! 

- 15 minute version of 45 minute talk
- how to best automate Foreman/Katello using Ansible
- spoiler: command: hammer is not the answer!
WHY NOT X?!

- `ansible-module-foreman` by Thomas Krahn (@Nosmoht) is probably the oldest
  - Well maintained
  - Supports only Foreman
- Upstream Ansible foreman and katello modules
  - Deprecated since Ansible 2.8
  - "one" module for everything
FOREMAN ANSIBLE MODULES

• Started June 2017 as a repository to clean up upstream modules
• One module per Foreman entity or action
• Extensive test-suite
• Abstraction framework for common tasks (connect, search, create, update, delete)
FOREMAN ANSIBLE MODULES

• Initially, we still used nailgun
  ▪ nailgun releases are Satellite version specific
  ▪ Plugins not in Satellite are not supported
  ▪ Doesn't work without Katello installed
• Recent switch to apypie
  ▪ Consumes the apidoc.json published by Foreman/apipie-rails
• Migration quite easy thanks to the existing framework and tests
FOREMAN ANSIBLE MODULES - STATS

- 43 🌟 on GitHub
- 24 Contributors (8 Red Hat, 7 ATIX)
- 8 new Contributors in 2019
Distribution of monthly PR activity
PR data split into old/new at 2019-02-01

Significant change in PR activity, p-value 0.047
FOREMAN ANSIBLE MODULES - OUTLOOK

- Collection on Ansible Galaxy
- RPM on yum.theforeman.org
LET'S WRITE A MODULE!
UNDER THE HOOD

Most modules manage objects/entities in Foreman

1. Find an existing entity
2. Compare existing entity with the data provided by the user
3. Save the entity

We have a framework to support that
First a wrapper around AnsibleModule:

```python
text = module = ForemanEntityApypieAnsibleModule(
    argument_spec=dict(name=dict(required=True)))
```
Load user provided parameters and connect to the API:

```
entity_dict = module.clean_params()
module.connect()
```
Find the entity and ensure it looks like the user wanted:

```python
entity = module.find_resource_by_name('architectures', name=entity_dict['name'], failsafe=True)
changed = module.ensure_resource_state('architectures', entity_dict, entity, name_map)
module.exit_json(changed=changed)
```
Translate Ansible params to Foreman API params:

```python
name_map = { 'name': 'name' }
```
from ansible.module_utils.foreman_helper import ForemanEntityApypieAnsibleModule

name_map = { 'name': 'name' }
module = ForemanEntityApypieAnsibleModule(
    argument_spec=dict(name=dict(required=True)))
entity_dict = module.clean_params()
module.connect()

entity = module.find_resource_by_name('architectures',
    name=entity_dict['name'], failsafe=True)
changed = module.ensure_resource_state('architectures',
    entity_dict, entity, name_map)
module.exit_json(changed=changed)
if not module.desired_absent:
    if 'operatingsystems' in entity_dict:
        entity_dict['operatingsystems'] =
            module.find_resources_by_title('operatingsystems',
            entity_dict['operatingsystems'], thin=True)
if not module.desired_absent:
    if 'operatingsystems' in entity_dict:
        search_list = ['title~{}'.format(title) for title in entity_dict['operatingsystems']]  
    entity_dict['operatingsystems'] =  
        module.find_resources('operatingsystems', search_list, thin=True)
THANKS!

evgeni@golov.de
die-welt.net
@zhenech
@zhenech@chaos.social
@evgeni
zhenech