

Managing Complexity

Config As Code, Custom Ansible Modules, and You

Zach Marine, Squarespace

About me



- Software engineer
- 3.5 years at Squarespace
- Focus on data tooling and infrastructure



- Platform for creating websites, domains, commerce stores, etc.
- “Squarespace makes beautiful products to help people with creative ideas succeed.”
- Millions of active subscriptions
- 650+ employees across New York, Portland, and Dublin

Agenda

1. Config As Code
2. Ansible
3. Testing

Agenda

Postgres
Permissions

- 
1. Config As Code
 2. Ansible
 3. Testing

Our Example: Postgres Permissions

Behavior

Effect

Our Example: Postgres Permissions

Behavior

Postgres permissions are highly flexible

Effect

Requires high level of knowledge

Similar “boilerplate” required for additional entities

Our Example: Postgres Permissions

Behavior

Postgres permissions are highly flexible

Effect

Requires high level of knowledge

Similar “boilerplate” required for additional entities

Info is distributed throughout the database

Hard to fully understand the setup

Our Example: Postgres Permissions

Behavior

Postgres permissions are highly flexible

Effect

Requires high level of knowledge

Similar “boilerplate” required for additional entities

Info is distributed throughout the database

Hard to fully understand the setup

Changes can be made live

No vetting from others

Testing environments may not accurately represent production

How can we make this better?

Our Example: Postgres Permissions

Behavior

Postgres permissions are highly flexible

Effect

Requires high level of knowledge

Similar “boilerplate” required for additional entities

Info is distributed throughout the database

Hard to fully understand the setup

Changes can be made live

No vetting from others

Testing environments may not accurately represent production

Our Example: Postgres Permissions

Behavior

Postgres permissions are highly flexible

Effect

Requires high level of knowledge

Simplify end user API

Similar ‘boilerplate’ required for additional entities

Info is distributed throughout the database

Hard to fully understand the setup

Changes can be made live

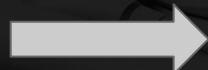
No vetting from others

Testing environments may not accurately represent production

Our Example: Postgres Permissions

Behavior

Postgres permissions are highly flexible



Effect

Requires high level of knowledge

Simplify end user API

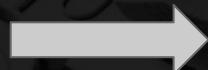
Similar ‘boilerplate’ required for additional entities

Info is distributed throughout the database



Hard to **Put all config together**

Changes can be made live



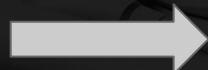
No vetting from others

Testing environments may not accurately represent production

Our Example: Postgres Permissions

Behavior

Postgres permissions are highly flexible



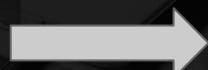
Effect

Requires high level of knowledge

Simplify end user API

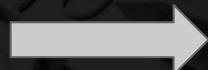
Similar ‘boilerplate’ required for additional entities

Info is distributed throughout the database



Hard to **Put all config together**

Changes can be made live



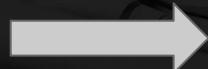
No **Get config into version control**

Testing environments may not accurately represent production

Our Example: Postgres Permissions

Behavior

Postgres permissions are highly flexible



Effect

Requires high level of knowledge

Simplify end user API

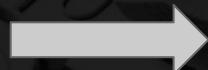
Similar ‘boilerplate’ required for additional entities

Info is distributed throughout the database



Hard to **Put all config together**

Changes can be made live



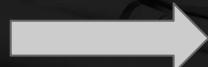
No **Get config into version control**

Testing environments may not accurately represent production
Make config deployable

Our Example: Postgres Permissions

Behavior

Postgres permissions are highly flexible



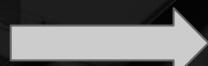
Effect

Requires high level of knowledge

Simplify end user API

Similar ‘boilerplate’ required for additional entities

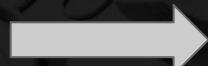
Info is distributed throughout the database



Hard to understand what’s what in setup

Put all config together

Changes can be made live



No **Get config into version control**

Testing environments may not accurately represent production

Make config deployable

What is config as code?

Enter: Config As Code

Definition

Putting application configuration into code

Motivation

Simplify our users' lives*

* Often ourselves!

Config As Code: Best Practices

Before

```
CREATE ROLE "svc-mlcompute" LOGIN;  
  
GRANT is_app TO "svc-mlcompute";  
  
GRANT USAGE ON SCHEMA dmart TO "svc-mlcompute";  
GRANT USAGE ON SCHEMA ods TO "svc-mlcompute";  
GRANT USAGE ON SCHEMA sqsp TO "svc-mlcompute";  
GRANT USAGE ON SCHEMA staging TO "svc-mlcompute";  
  
GRANT SELECT ON ALL TABLES IN SCHEMA dmart TO "svc-mlcompute";  
GRANT SELECT ON ALL TABLES IN SCHEMA ods TO "svc-mlcompute";  
GRANT SELECT ON ALL TABLES IN SCHEMA sqsp TO "svc-mlcompute";  
GRANT SELECT ON ALL TABLES IN SCHEMA staging TO "svc-mlcompute";  
  
ALTER DEFAULT PRIVILEGES IN SCHEMA dmart GRANT SELECT ON TABLES TO "svc-mlcompute";  
ALTER DEFAULT PRIVILEGES IN SCHEMA ods GRANT SELECT ON TABLES TO "svc-mlcompute";  
ALTER DEFAULT PRIVILEGES IN SCHEMA sqsp GRANT SELECT ON TABLES TO "svc-mlcompute";  
ALTER DEFAULT PRIVILEGES IN SCHEMA staging GRANT SELECT ON TABLES TO "svc-mlcompute";
```

After

```
_name: svc-mlcompute  
can_login: yes  
membership:  
  - is_app  
privileges:  
  read:  
    schema:  
      - dmart  
      - ods  
      - sqsp  
      - staging  
    table:  
      - dmart.*  
      - ods.*  
      - sqsp.*  
      - staging.*
```

Config As Code: Best Practices

1. Simplify our users' lives by using terminology they understand.

Before

```
CREATE ROLE "svc-mlcompute" LOGIN;  
  
GRANT is_app TO "svc-mlcompute";  
  
GRANT USAGE ON SCHEMA dmart TO "svc-mlcompute";  
GRANT USAGE ON SCHEMA ods TO "svc-mlcompute";  
GRANT USAGE ON SCHEMA sqsp TO "svc-mlcompute";  
GRANT USAGE ON SCHEMA staging TO "svc-mlcompute";  
  
GRANT SELECT ON ALL TABLES IN SCHEMA dmart TO "svc-mlcompute";  
GRANT SELECT ON ALL TABLES IN SCHEMA ods TO "svc-mlcompute";  
GRANT SELECT ON ALL TABLES IN SCHEMA sqsp TO "svc-mlcompute";  
GRANT SELECT ON ALL TABLES IN SCHEMA staging TO "svc-mlcompute";  
  
ALTER DEFAULT PRIVILEGES IN SCHEMA dmart GRANT SELECT ON TABLES TO "svc-mlcompute";  
ALTER DEFAULT PRIVILEGES IN SCHEMA ods GRANT SELECT ON TABLES TO "svc-mlcompute";  
ALTER DEFAULT PRIVILEGES IN SCHEMA sqsp GRANT SELECT ON TABLES TO "svc-mlcompute";  
ALTER DEFAULT PRIVILEGES IN SCHEMA staging GRANT SELECT ON TABLES TO "svc-mlcompute";
```

After

```
_name: svc-mlcompute  
can_login: yes  
membership:  
  - is_app  
privileges:  
  - read:  
    schema:  
      - dmart  
      - ods  
      - sqsp  
      - staging  
    table:  
      - dmart.*  
      - ods.*  
      - sqsp.*  
      - staging.*
```

Config As Code: Best Practices

1. Simplify our users' lives by using terminology they understand.

Before

```
CREATE ROLE "svc-mlcompute" LOGIN;  
  
GRANT is_app TO "svc-mlcompute";  
  
GRANT USAGE ON SCHEMA dmart TO "svc-mlcompute";  
GRANT USAGE ON SCHEMA ods TO "svc-mlcompute";  
GRANT USAGE ON SCHEMA sqsp TO "svc-mlcompute";  
GRANT USAGE ON SCHEMA staging TO "svc-mlcompute";  
  
GRANT SELECT ON ALL TABLES IN SCHEMA dmart TO "svc-mlcompute";  
GRANT SELECT ON ALL TABLES IN SCHEMA ods TO "svc-mlcompute";  
GRANT SELECT ON ALL TABLES IN SCHEMA sqsp TO "svc-mlcompute";  
GRANT SELECT ON ALL TABLES IN SCHEMA staging TO "svc-mlcompute";  
  
ALTER DEFAULT PRIVILEGES IN SCHEMA dmart GRANT SELECT ON TABLES TO "svc-mlcompute";  
ALTER DEFAULT PRIVILEGES IN SCHEMA ods GRANT SELECT ON TABLES TO "svc-mlcompute";  
ALTER DEFAULT PRIVILEGES IN SCHEMA sqsp GRANT SELECT ON TABLES TO "svc-mlcompute";  
ALTER DEFAULT PRIVILEGES IN SCHEMA staging GRANT SELECT ON TABLES TO "svc-mlcompute";
```

Database -> CONNECT

After

```
_name: svc-mlcompute  
can_login: yes  
membership:  
  -_is_app  
privileges:  
  - read:  
    schema:  
      -_dmart  
      -_ods  
      -_sqsp  
      -_staging  
    table:  
      -_dmart.*  
      -_ods.*  
      -_sqsp.*  
      -_staging.*
```

Config As Code: Best Practices

1. Simplify our users' lives by using terminology they understand.

Before

```
CREATE ROLE "svc-mlcompute" LOGIN;  
  
GRANT is_app TO "svc-mlcompute";  
  
GRANT USAGE ON SCHEMA dmart TO "svc-mlcompute";  
GRANT USAGE ON SCHEMA ods TO "svc-mlcompute";  
GRANT USAGE ON SCHEMA sqsp TO "svc-mlcompute";  
GRANT USAGE ON SCHEMA staging TO "svc-mlcompute";  
  
GRANT SELECT ON ALL TABLES IN SCHEMA dmart TO "svc-mlcompute";  
GRANT SELECT ON ALL TABLES IN SCHEMA ods TO "svc-mlcompute";  
GRANT SELECT ON ALL TABLES IN SCHEMA sqsp TO "svc-mlcompute";  
GRANT SELECT ON ALL TABLES IN SCHEMA staging TO "svc-mlcompute";  
  
ALTER DEFAULT PRIVILEGES IN SCHEMA dmart GRANT SELECT ON TABLES TO "svc-mlcompute";  
ALTER DEFAULT PRIVILEGES IN SCHEMA ods GRANT SELECT ON TABLES TO "svc-mlcompute";  
ALTER DEFAULT PRIVILEGES IN SCHEMA sqsp GRANT SELECT ON TABLES TO "svc-mlcompute";  
ALTER DEFAULT PRIVILEGES IN SCHEMA staging GRANT SELECT ON TABLES TO "svc-mlcompute";
```

Database -> CONNECT

After

Sequence -> SELECT

```
name: svc-mlcompute  
  login: yes  
  
privileges:  
  read:  
    schema:  
      - dmart  
      - ods  
      - sqsp  
      - staging  
    table:  
      - dmart.*  
      - ods.*  
      - sqsp.*  
      - staging.*
```

Config As Code: Best Practices

Foreign Server -> USAGE

- Simply, our users lives by the terminology they understand.

Before

```
CREATE ROLE "svc-mlcompute" LOGIN;  
  
GRANT is_app TO "svc-mlcompute";  
  
GRANT USAGE ON SCHEMA dmart TO "svc-mlcompute";  
GRANT USAGE ON SCHEMA ods TO "svc-mlcompute";  
GRANT USAGE ON SCHEMA sqsp TO "svc-mlcompute";  
GRANT USAGE ON SCHEMA staging TO "svc-mlcompute";  
  
GRANT SELECT ON ALL TABLES IN SCHEMA dmart TO "svc-mlcompute";  
GRANT SELECT ON ALL TABLES IN SCHEMA ods TO "svc-mlcompute";  
GRANT SELECT ON ALL TABLES IN SCHEMA sqsp TO "svc-mlcompute";  
GRANT SELECT ON ALL TABLES IN SCHEMA staging TO "svc-mlcompute";  
  
ALTER DEFAULT PRIVILEGES IN SCHEMA dmart GRANT SELECT ON TABLES TO "svc-mlcompute";  
ALTER DEFAULT PRIVILEGES IN SCHEMA ods GRANT SELECT ON TABLES TO "svc-mlcompute";  
ALTER DEFAULT PRIVILEGES IN SCHEMA sqsp GRANT SELECT ON TABLES TO "svc-mlcompute";  
ALTER DEFAULT PRIVILEGES IN SCHEMA staging GRANT SELECT ON TABLES TO "svc-mlcompute";
```

Database -> CONNECT

Sequence -> SELECT

```
name: svc-mlcompute  
  login: yes  
  
privileges:  
  read:  
    schema:  
      - dmart  
      - ods  
      - sqsp  
      - staging  
    table:  
      - dmart.*  
      - ods.*  
      - sqsp.*  
      - staging.*
```

Config As Code: Best Practices

Foreign Server -> USAGE

- Simply, our users lives by the terminology they understand.

Before

```

CREATE ROLE "svc-mlcompute" LOGIN;

GRANT is_app TO "svc-mlcompute";

GRANT USAGE ON SCHEMA dmart TO "svc-mlcompute";
GRANT USAGE ON SCHEMA ods TO "svc-mlcompute";
GRANT USAGE ON SCHEMA sqsp TO "svc-mlcompute";
GRANT USAGE ON SCHEMA staging TO "svc-mlcompute";

GRANT SELECT ON ALL TABLES IN SCHEMA dmart TO "svc-mlcompute";
GRANT SELECT ON ALL TABLES IN SCHEMA ods TO "svc-mlcompute";
GRANT SELECT ON ALL TABLES IN SCHEMA sqsp TO "svc-mlcompute";
GRANT SELECT ON ALL TABLES IN SCHEMA staging TO "svc-mlcompute";

ALTER DEFAULT PRIVILEGES IN SCHEMA dmart GRANT SELECT ON TABLES TO "svc-mlcompute";
ALTER DEFAULT PRIVILEGES IN SCHEMA ods GRANT SELECT ON TABLES TO "svc-mlcompute";
ALTER DEFAULT PRIVILEGES IN SCHEMA sqsp GRANT SELECT ON TABLES TO "svc-mlcompute";
ALTER DEFAULT PRIVILEGES IN SCHEMA staging GRANT SELECT ON TABLES TO "svc-mlcompute";

```

Database -> CONNECT

Large Object -> SELECT

After

Sequence -> SELECT

```

name: svc-mlcompute
  login: yes
  privs:
    read:
      schema:
        - dmart
        - ods
        - sqsp
        - staging
      table:
        - dmart.*
        - ods.*
        - sqsp.*
        - staging.*

```

Config As Code: Best Practices

Foreign Server -> USAGE

- Simply, our users lives by using terminology they understand.

Before

```

CREATE ROLE "svc-mlcompute" LOGIN;

GRANT is_app TO "svc-mlcompute";

GRANT USAGE ON SCHEMA dmart TO "svc-mlcompute";
GRANT USAGE ON SCHEMA ods TO "svc-mlcompute";
GRANT USAGE ON SCHEMA sqsp TO "svc-mlcompute";
GRANT USAGE ON SCHEMA staging TO "svc-mlcompute";

GRANT SELECT ON ALL TABLES IN SCHEMA dmart TO "svc-mlcompute";
GRANT SELECT ON ALL TABLES IN SCHEMA ods TO "svc-mlcompute";
GRANT SELECT ON ALL TABLES IN SCHEMA sqsp TO "svc-mlcompute";
GRANT SELECT ON ALL TABLES IN SCHEMA staging TO "svc-mlcompute";

ALTER DEFAULT PRIVILEGES IN SCHEMA dmart GRANT SELECT ON TABLES TO "svc-mlcompute";
ALTER DEFAULT PRIVILEGES IN SCHEMA ods GRANT SELECT ON TABLES TO "svc-mlcompute";
ALTER DEFAULT PRIVILEGES IN SCHEMA sqsp GRANT SELECT ON TABLES TO "svc-mlcompute";
ALTER DEFAULT PRIVILEGES IN SCHEMA staging GRANT SELECT ON TABLES TO "svc-mlcompute";

```

Database -> CONNECT

Large Object -> SELECT

After

Sequence -> SELECT

Language -> USAGE

```

name: svc-mlcompute
  login: yes
  privs:
    - read
  schema:
    - dmart
    - ods
    - sqsp
    - tagging
    - tables
    - dmart.*
    - ods.*
    - sqsp.*
    - staging.*

```

Config As Code: Best Practices

Foreign Server -> **READ**

1. Simplify our users lives by using terminology they understand.

Before

```
CREATE ROLE "svc-mlcompute" LOGIN;

GRANT is_app TO "svc-mlcompute";

GRANT USAGE ON SCHEMA dmart TO "svc-mlcompute";
GRANT USAGE ON SCHEMA ods TO "svc-mlcompute";
GRANT USAGE ON SCHEMA sqsp TO "svc-mlcompute";
GRANT USAGE ON SCHEMA staging TO "svc-mlcompute";

GRANT SELECT ON ALL TABLES IN SCHEMA dmart TO "svc-mlcompute";
GRANT SELECT ON ALL TABLES IN SCHEMA ods TO "svc-mlcompute";
GRANT SELECT ON ALL TABLES IN SCHEMA sqsp TO "svc-mlcompute";
GRANT SELECT ON ALL TABLES IN SCHEMA staging TO "svc-mlcompute";

ALTER DEFAULT PRIVILEGES IN SCHEMA dmart GRANT SELECT ON TABLES TO "svc-mlcompute";
ALTER DEFAULT PRIVILEGES IN SCHEMA ods GRANT SELECT ON TABLES TO "svc-mlcompute";
ALTER DEFAULT PRIVILEGES IN SCHEMA sqsp GRANT SELECT ON TABLES TO "svc-mlcompute";
ALTER DEFAULT PRIVILEGES IN SCHEMA staging GRANT SELECT ON TABLES TO "svc-mlcompute";
```

Database -> **READ**

After

Sequence -> **READ**

```
name: svc-mlcompute
  login: yes
```

```
privileges:
  read:
```

```
schema:
  - dmart
  - ods
  - sqsp
  - tagging
  tables:
    - dmart.*
```

Language -> **READ**

```
- ods.*
```

```
- sqsp.*
```

```
- staging.*
```

Large Object -> **READ**

Config As Code: Best Practices

2. Simplify our users' lives by reusing techniques they're familiar with.

Before

```
CREATE ROLE "svc-mlcompute" LOGIN;  
  
GRANT is_app TO "svc-mlcompute";  
  
GRANT USAGE ON SCHEMA dmart TO "svc-mlcompute";  
GRANT USAGE ON SCHEMA ods TO "svc-mlcompute";  
GRANT USAGE ON SCHEMA sqsp TO "svc-mlcompute";  
GRANT USAGE ON SCHEMA staging TO "svc-mlcompute";  
  
GRANT SELECT ON ALL TABLES IN SCHEMA dmart TO "svc-mlcompute";  
GRANT SELECT ON ALL TABLES IN SCHEMA ods TO "svc-mlcompute";  
GRANT SELECT ON ALL TABLES IN SCHEMA sqsp TO "svc-mlcompute";  
GRANT SELECT ON ALL TABLES IN SCHEMA staging TO "svc-mlcompute";  
  
ALTER DEFAULT PRIVILEGES IN SCHEMA dmart GRANT SELECT ON TABLES TO "svc-mlcompute";  
ALTER DEFAULT PRIVILEGES IN SCHEMA ods GRANT SELECT ON TABLES TO "svc-mlcompute";  
ALTER DEFAULT PRIVILEGES IN SCHEMA sqsp GRANT SELECT ON TABLES TO "svc-mlcompute";  
ALTER DEFAULT PRIVILEGES IN SCHEMA staging GRANT SELECT ON TABLES TO "svc-mlcompute";
```

After

```
_name: svc-mlcompute  
can_login: yes  
membership:  
  - is_app  
privileges:  
  read:  
    schema:  
      - dmart  
      - ods  
      - sqsp  
      - staging  
    table:  
      - dmart.*  
      - ods.*  
      - sqsp.*  
      - staging.*
```

Config As Code: Best Practices

3. Simplify our users' lives by abstracting away implementation details.

Before

```
CREATE ROLE "svc-mlcompute" LOGIN;  
  
GRANT is_app TO "svc-mlcompute";  
  
GRANT USAGE ON SCHEMA dmart TO "svc-mlcompute";  
GRANT USAGE ON SCHEMA ods TO "svc-mlcompute";  
GRANT USAGE ON SCHEMA sqsp TO "svc-mlcompute";  
GRANT USAGE ON SCHEMA staging TO "svc-mlcompute";  
  
GRANT SELECT ON ALL TABLES IN SCHEMA dmart TO "svc-mlcompute";  
GRANT SELECT ON ALL TABLES IN SCHEMA ods TO "svc-mlcompute";  
GRANT SELECT ON ALL TABLES IN SCHEMA sqsp TO "svc-mlcompute";  
GRANT SELECT ON ALL TABLES IN SCHEMA staging TO "svc-mlcompute";  
  
ALTER DEFAULT PRIVILEGES IN SCHEMA dmart GRANT SELECT ON TABLES TO "svc-mlcompute";  
ALTER DEFAULT PRIVILEGES IN SCHEMA ods GRANT SELECT ON TABLES TO "svc-mlcompute";  
ALTER DEFAULT PRIVILEGES IN SCHEMA sqsp GRANT SELECT ON TABLES TO "svc-mlcompute";  
ALTER DEFAULT PRIVILEGES IN SCHEMA staging GRANT SELECT ON TABLES TO "svc-mlcompute";
```

After

```
_name: svc-mlcompute  
can_login: yes  
membership:  
  -_is_app  
privileges:  
  read:  
    schema:  
      -_dmart  
      -_ods  
      -_sqsp  
      -_staging  
    table:  
      -_dmart.*  
      -_ods.*  
      -_sqsp.*  
      -_staging.*
```

Config As Code: Best Practices

4. Simplify our users' lives by removing boilerplate.

Before

```
CREATE ROLE "svc-mlcompute" LOGIN;  
  
GRANT is_app TO "svc-mlcompute";  
  
GRANT USAGE ON SCHEMA dmart TO "svc-mlcompute";  
GRANT USAGE ON SCHEMA ods TO "svc-mlcompute";  
GRANT USAGE ON SCHEMA sqsp TO "svc-mlcompute";  
GRANT USAGE ON SCHEMA staging TO "svc-mlcompute";  
  
GRANT SELECT ON ALL TABLES IN SCHEMA dmart TO "svc-mlcompute";  
GRANT SELECT ON ALL TABLES IN SCHEMA ods TO "svc-mlcompute";  
GRANT SELECT ON ALL TABLES IN SCHEMA sqsp TO "svc-mlcompute";  
GRANT SELECT ON ALL TABLES IN SCHEMA staging TO "svc-mlcompute";  
  
ALTER DEFAULT PRIVILEGES IN SCHEMA dmart GRANT SELECT ON TABLES TO "svc-mlcompute";  
ALTER DEFAULT PRIVILEGES IN SCHEMA ods GRANT SELECT ON TABLES TO "svc-mlcompute";  
ALTER DEFAULT PRIVILEGES IN SCHEMA sqsp GRANT SELECT ON TABLES TO "svc-mlcompute";  
ALTER DEFAULT PRIVILEGES IN SCHEMA staging GRANT SELECT ON TABLES TO "svc-mlcompute";
```

After

```
_name: svc-mlcompute  
can_login: yes  
membership:  
  -_is_app  
privileges:  
  read:  
    schema:  
      -_dmart  
      -_ods  
      -_sqsp  
      -_staging  
    table:  
      -_dmart.*  
      -_ods.*  
      -_sqsp.*  
      -_staging.*
```

⌚ YAML (“Yet Another Markup Language”)

We chose to put our config into YAML

Benefits for our users:

- Familiar to many users
- Not a lot of boilerplate
 - Easy to read
 - Easy to write

Benefits for ourselves:

- Easy to parse in Python

```
- name: service_account1
  can_login: yes
  membership:
    - is_app
  privileges:
    read:
      schema:
        - metadata
      table:
        - metadata.oplog_tail_status

- name: person1
  can_login: yes
  has_personal_schema: yes
  membership:
    - analyst

- name: zmarine
  can_login: yes
  has_personal_schema: yes
  membership:
    - data-engineer
```

YAML (“Yet Another Markup Language”)

```
(e) code In [1] import yaml
```

```
with open('spec.yml', 'r') as f:  
    spec = yaml.load(f)
```

```
(e) code In [2] from pprint import pprint
```

```
pprint(spec)
```

```
[{'can_login': True,  
 'membership': ['is_app'],  
 'name': 'service_account1',  
 'privileges': {'read': {'schema': ['metadata'],  
 'table': ['metadata.oplog_tail_status']}},  
 {'can_login': True,  
 'has_personal_schema': True,  
 'membership': ['analyst'],  
 'name': 'person1'},  
 {'can_login': True,  
 'has_personal_schema': True,  
 'membership': ['data-engineer'],  
 'name': 'zmarine'}]
```

```
_name: service_account1  
can_login: yes  
membership:  
    -_is_app  
privileges:  
    read:  
        schema:  
            -_metadata  
        table:  
            -_metadata.oplog_tail_status  
  
_name: person1  
can_login: yes  
has_personal_schema: yes  
membership:  
    -_analyst  
  
_name: zmarine  
can_login: yes  
has_personal_schema: yes  
membership:  
    -_data-engineer
```

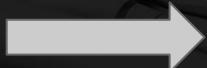
Great! We've got our config into code.

...Now what?

Our Example: Postgres Permissions

Behavior

Postgres permissions are highly flexible



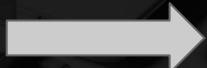
Effect

Requires high level of knowledge

Simplify end user API

Similar ‘boilerplate’ required for additional entities

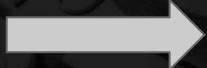
Info is distributed throughout the database



Hard to understand what’s what in setup

Put all config together

Changes can be made live



No **Get config into version control**

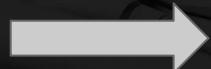
Testing environments may not accurately represent production

Make config deployable

Our Example: Postgres Permissions

Behavior

Postgres permissions are highly flexible



Effect

Requires high level of knowledge

Simplify end user API

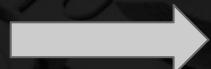
Similar ‘boilerplate’ required for additional entities

Info is distributed throughout the database



Hard to **Put all config together**

Changes can be made live



No **Get config into version control**

Testing environments may not accurately represent production
Make config deployable

What is ansible?

What is Ansible?



ANSIBLE

Ansible described itself as:

“A radically simple IT automation engine”

Often described as “infrastructure as code”

Similar tools:

- Chef
- Puppet
- SaltStack
- Terraform

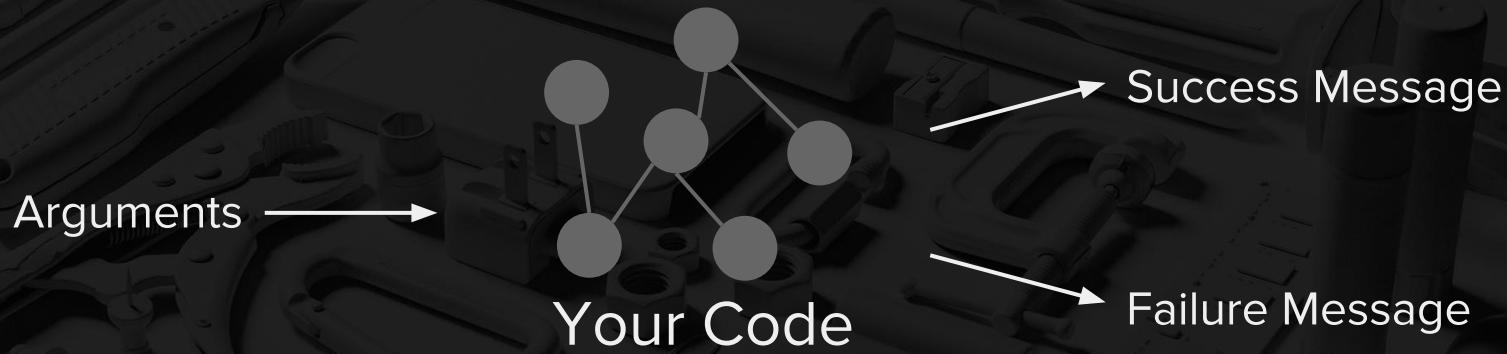
Why Ansible?

- Currently the automation tool of choice at Squarespace
- Easy to write your own module
- Supports secure secret storage
- Modules typically support “check” mode
(see what will change without actually doing it)

Why Ansible?

- Currently the automation tool of choice at Squarespace
- Easy to write your own module
- Supports secure secret storage
- Modules typically support “check” mode
(see what will change without actually doing it)

Ansible And Custom Modules



Ansible And Custom Modules

Ansible

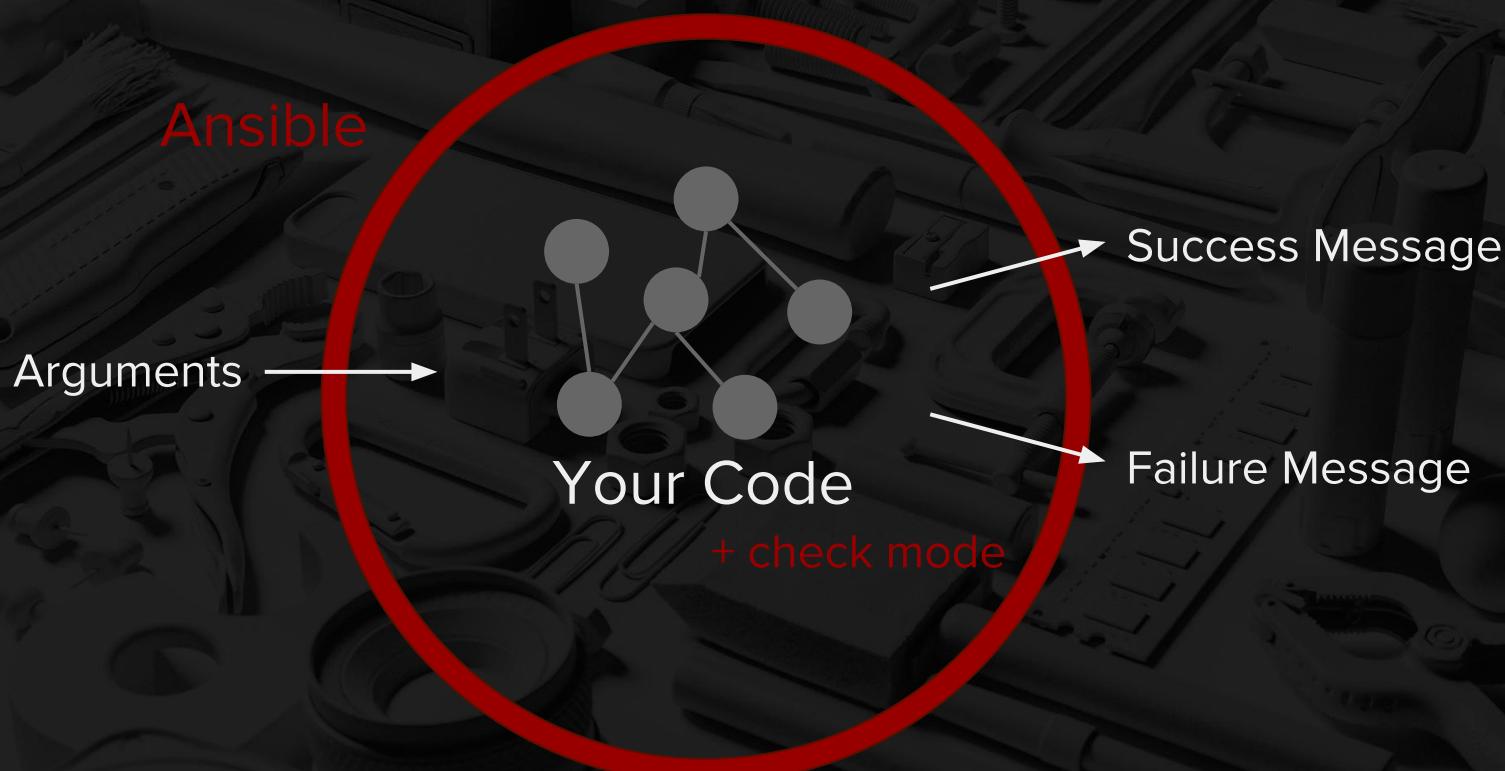
Arguments



Success Message

Failure Message

Ansible And Custom Modules



Ansible And Custom Modules

Ansible

Arguments
name: "Hello-World"



+ check mode

Success Message
"Hello-World"

Failure Message

Ansible Execution

You tell ansible
to run playbook

Ansible tells
module to run

Module runs

Output is
produced



Ansible Execution



```
>>> pip install ansible  
>>> export ANSIBLE_LIBRARY='.'  
>>> ansible-playbook -v basic_playbook.yml
```

Ansible Execution



1 ./basic_playbook.yml

Servers to run against

```
1 ---  
2 - name: A basic playbook  
3 -> hosts: localhost  
4 tasks:  
5 -> - name: Test that my module works  
6 -> basic_module:  
7 -> name: "Hello-World"
```

Description of task

Module name

Module arguments

Ansible Execution

You tell ansible
to run playbook

Ansible tells
module to run

Module runs

Output is produced

```
./basic_module.py
1 #!/usr/bin/python
2
3 from ansible.module_utils.basic import AnsibleModule
4
5
6 def setup_module():
7     return AnsibleModule(
8         argument_spec=dict(
9             name=dict(required=True, type='str'),
10            ),
11            supports_check_mode=True
12        )
13
14
15 def main():
16     module = setup_module()
17     module.exit_json(changed=True, msg=module.params['name'])
18
19
20 if __name__ == '__main__':
21     main()
22
```

Ansible Execution



Modules don't have to be Python!

```
./basic_module.py
1 #!/usr/bin/python
2
3 from ansible.module_utils.basic import AnsibleModule
4
5
6 def setup_module():
7     return AnsibleModule(
8         argument_spec=dict(
9             name=dict(required=True, type='str'),
10            ),
11            supports_check_mode=True
12        )
13
14
15 def main():
16     module = setup_module()
17     module.exit_json(changed=True, msg=module.params['name'])
18
19
20 if __name__ == '__main__':
21     main()
22
```

Ansible Execution



Define expected arguments

```
./basic_module.py
1 #!/usr/bin/python
2
3 from ansible.module_utils.basic import AnsibleModule
4
5
6 def setup_module():
7     return AnsibleModule(
8         argument_spec=dict(
9             name=dict(required=True, type='str'),
10            ),
11            supports_check_mode=True
12        )
13
14
15 def main():
16     module = setup_module()
17     module.exit_json(changed=True, msg=module.params['name'])
18
19
20 if __name__ == '__main__':
21     main()
22
```

Ansible Execution



```
./basic_module.py
1 #!/usr/bin/python
2
3 from ansible.module_utils.basic import AnsibleModule
4
5
6 def setup_module():
7     return AnsibleModule(
8         argument_spec=dict(
9             name=dict(required=True, type='str'),
10            ),
11            supports_check_mode=True
12        )
13
14
15 def main():
16     module = setup_module()
17     module.exit_json(changed=True, msg=module.params['name'])
18
19
20 if __name__ == '__main__':
21     main()
22
```

Provide info upon exit



Ansible Execution



```
PLAY [localhost] *****
TASK [Gathering Facts] *****
ok: [localhost]

TASK [Test that my module works] *****
changed: [localhost] => {"changed": true, "failed": false, "msg": "Hello-World"}

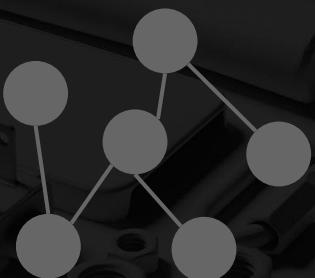
PLAY RECAP *****
localhost : ok=2      changed=1      unreachable=0      failed=0
```

Ansible Execution



Ansible

Arguments



Your Code

+ check mode

Success Message

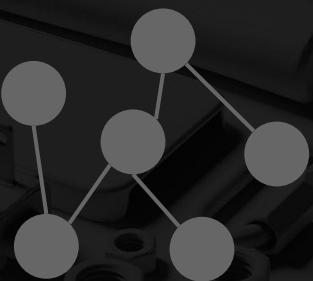
Failure Message

Ansible Execution



Ansible

Arguments
module.params



Your Code
+ check mode

Success Message
module.exit_json()

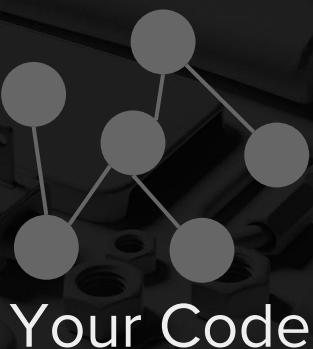
Failure Message

Ansible Execution



Ansible

Arguments
module.params



+ check mode

Success Message
module.exit_json()

Failure Message

Ansible Execution



Ansible

Arguments
`module.params`



Your Code
+ check mode
`module.check_mode`

Success Message
`module.exit_json()`

Failure Message
`module.fail_json()`

Ansible Execution: fail_json + check mode

```
14
15 def main():
16     module = setup_module()
17
18     if module.check_mode:
19         module.exit_json(changed=False, msg='This is check mode')
20
21     if module.params['name'] == 'Oscar':
22         module.fail_json(msg="Don't use the name 'Oscar'")
23
24     module.exit_json(changed=False, msg=module.params['name'])
25
```

```
>>> ansible-playbook -v basic_playbook.yml --check
```

```
PLAY [localhost] ****
TASK [Gathering Facts] ****
ok: [localhost]

TASK [Test that my module works] ****
ok: [localhost] => {"changed": false, "failed": false, "msg": "This is check mode"}

PLAY RECAP ****
localhost          : ok=2      changed=0      unreachable=0      failed=0
```

Ansible Execution: fail_json + check mode

```
14
15 def main():
16     module = setup_module()
17
18     if module.check_mode:
19         module.exit_json(changed=False, msg='This is check mode')
20
21     if module.params['name'] == 'Oscar':
22         module.fail_json(msg="Don't use the name 'Oscar'")
23
24     module.exit_json(changed=False, msg=module.params['name'])
25
```

```
>>> ansible-playbook -v basic_playbook.yml
```

```
PLAY [localhost] ****
TASK [Gathering Facts] ****
ok: [localhost]

TASK [Test that my module works] ****
fatal: [localhost]: FAILED! => {"changed": false, "failed": true, "msg": "Don't use the name 'Oscar'"}

PLAY RECAP ****
localhost                  : ok=1      changed=0      unreachable=0      failed=1
```

Ansible Execution



Ansible

Arguments
`module.params`



Your Code
+ check mode
`module.check_mode`

Success Message
`module.exit_json()`

Failure Message
`module.fail_json()`

Testing

- [x] Config as code
- [x] Ansible
- [?] Testing

Testing Goals

1. Assert that our ansible module works as expected

Testing Goals

1. Assert that our ansible module works as expected
2. Keep tests isolated from real systems

Testing Goals

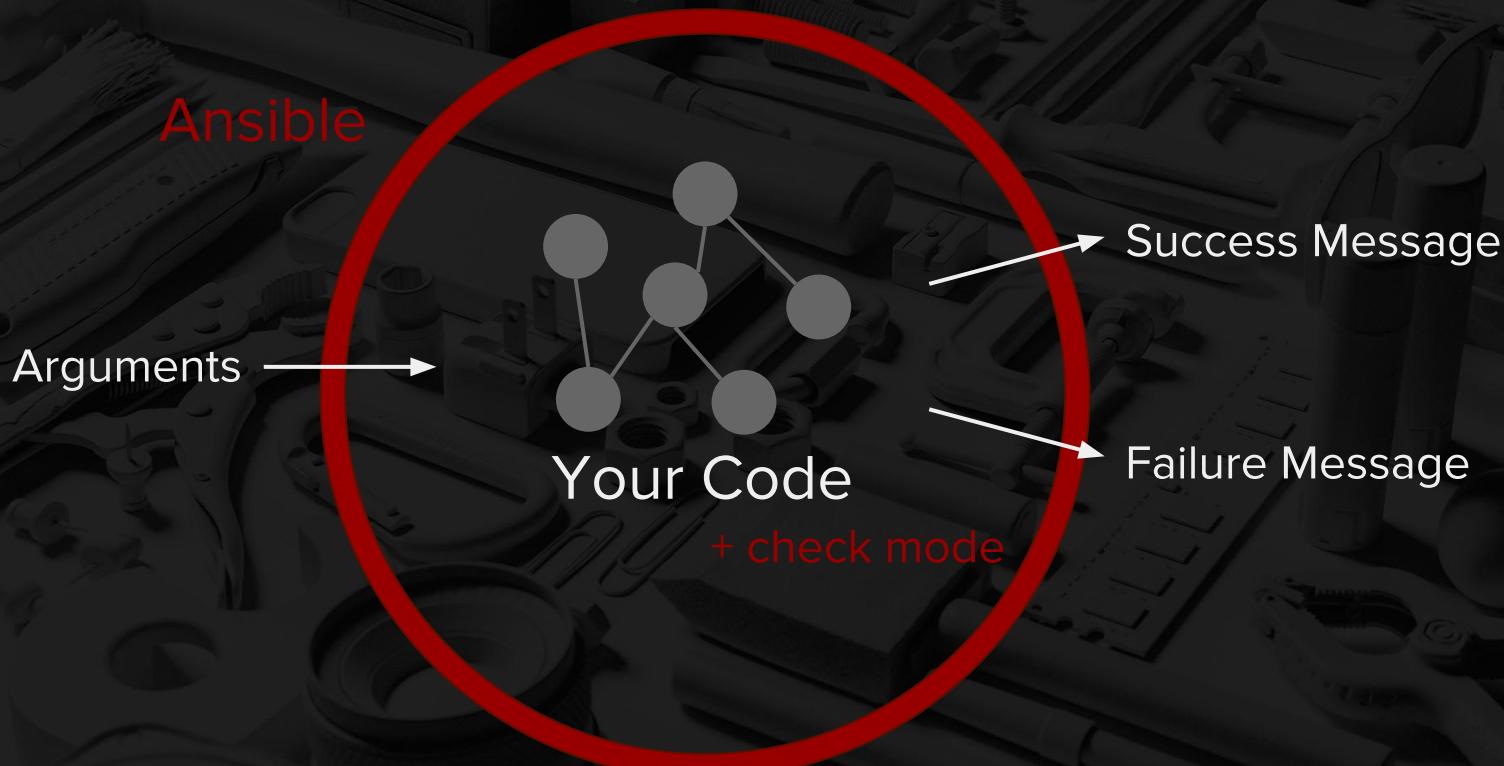
1. Assert that our ansible module works as expected
2. Keep tests isolated from real systems
3. Make testing both Python 2 and Python 3 possible

Testing Goals

1. Assert that our ansible module works as expected
2. Keep tests isolated from real systems
3. Make testing both Python 2 and Python 3 possible

Testing Implementation

1. Assert that our ansible module works as expected



Testing Implementation

1. Assert that our ansible module works as expected



Testing Implementation

1. Assert that our ansible module works as expected



Testing Implementation

Mock fail_json to log its output: *

```
44 @pytest.fixture(scope='session')
45 def mockmodule():
46
47     class MockAnsibleModule(object):
48         def fail_json(self, msg):
49             logging.basicConfig()
50             logging.error(msg)
51
52     return MockAnsibleModule()
```

Use the ‘caplog’ fixture from the pytest-catchlog plugin:

```
16 def test_init_errors_on_single_quotes_in_name(caplog, mockmodule, cursor):
17     role = {'name': "bad'_name"}
18     _ = pgb._BaseConfigurer(mockmodule, cursor, role)
19     assert caplog.records[0].msg == 'Role "bad\'_name" contains an unsupported character: \' or ''
```

* Other mocked methods not shown

Testing Implementation

1. Assert that our ansible module works as expected
2. Keep tests isolated from real systems
3. Make testing both Python 2 and Python 3 possible

Testing Implementation

1. Assert that our ansible module works as expected
2. Keep tests isolated from real systems
3. Make testing both Python 2 and Python 3 possible



```
Makefile
1 .PHONY: clean build clean create network remove network sleep start postgres stop
2
3 DEPS := $(shell env)
4 COMPOSE_NETWORK := $(patsubst %,network,$(COMPOSE_PROJECT_NAME))
5 POSTGRES_DB := $(patsubst %,db,$(COMPOSE_PROJECT_NAME))
6 POSTGRES_USER := $(patsubst %,user,$(COMPOSE_PROJECT_NAME))
7 POSTGRES_PASSWORD := test_password
8
9 attach:
10     docker exec -it $(POSTGRES_DB) psql -d $(POSTGRES_DB) -U $(POSTGRES_USER)
11
12 build:
13     docker build -t tester2 .
14     docker build -t tester3 .
15
16 .PHONY: test
17 test:
18     docker run --rm -it $(POSTGRES_DB) psql -d $(POSTGRES_DB) -U $(POSTGRES_USER) -c "CREATE SCHEMA IF NOT EXISTS test"
19     docker run --rm -it $(POSTGRES_DB) psql -d $(POSTGRES_DB) -U $(POSTGRES_USER) -c "CREATE TABLE IF NOT EXISTS test.test_table (id INT);"
20
21 .PHONY: clean
22 clean:
23     docker rm -f $(POSTGRES_DB)
24
25 .PHONY: create
26 create:
27     docker network create $(COMPOSE_NETWORK)
28
29 .PHONY: remove
30 remove:
31     docker network rm $(COMPOSE_NETWORK)
```

Makefile

Testing Implementation

1. Assert that our ansible module works as expected
2. Keep tests isolated from real systems
3. Make testing both Python 2 and Python 3 possible

Why don't you just use tox??

```
#!/usr/bin/env bash
# This file is part of https://github.com/PyCQA/pytest-docker
# Copyright (c) 2015-2016, PyCQA
# All rights reserved.
#
# Redistribution and use in source and binary forms, with or without
# modification, are permitted provided that the following conditions are met:
#
# * Redistributions of source code must retain the above copyright notice, this
#   list of conditions and the following disclaimer.
#
# * Redistributions in binary form must reproduce the above copyright notice,
#   this list of conditions and the following disclaimer in the documentation
#   and/or other materials provided with the distribution.
#
# * Neither the name of PyCQA nor the names of its contributors may be used to
#   endorse or promote products derived from this software without specific
#   prior written permission.
#
# THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
# AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
# IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
# DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE
# FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
# DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
# SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
# CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
# OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
# OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

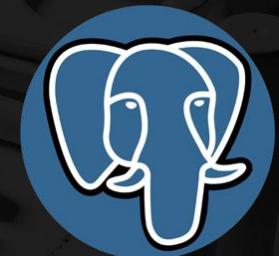
# This file is part of https://github.com/PyCQA/pytest-docker
# Copyright (c) 2015-2016, PyCQA
# All rights reserved.
#
# Redistribution and use in source and binary forms, with or without
# modification, are permitted provided that the following conditions are met:
#
# * Redistributions of source code must retain the above copyright notice, this
#   list of conditions and the following disclaimer.
#
# * Redistributions in binary form must reproduce the above copyright notice,
#   this list of conditions and the following disclaimer in the documentation
#   and/or other materials provided with the distribution.
#
# * Neither the name of PyCQA nor the names of its contributors may be used to
#   endorse or promote products derived from this software without specific
#   prior written permission.
#
# THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
# AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
# IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
# DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE
# FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
# DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
# SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
# CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
# OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
# OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

# This file is part of https://github.com/PyCQA/pytest-docker
# Copyright (c) 2015-2016, PyCQA
# All rights reserved.
#
# Redistribution and use in source and binary forms, with or without
# modification, are permitted provided that the following conditions are met:
#
# * Redistributions of source code must retain the above copyright notice, this
#   list of conditions and the following disclaimer.
#
# * Redistributions in binary form must reproduce the above copyright notice,
#   this list of conditions and the following disclaimer in the documentation
#   and/or other materials provided with the distribution.
#
# * Neither the name of PyCQA nor the names of its contributors may be used to
#   endorse or promote products derived from this software without specific
#   prior written permission.
#
# THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
# AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
# IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
# DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE
# FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
# DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
# SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
# CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
# OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
# OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

Testing Implementation

1. Assert that our ansible module works as expected
2. Keep tests isolated from real systems
3. Make testing both Python 2 and Python 3 possible

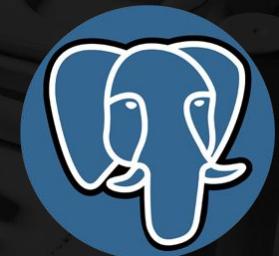
#1) Spin up a new postgres database



Testing Implementation

1. Assert that our ansible module works as expected
2. Keep tests isolated from real systems
3. Make testing both Python 2 and Python 3 possible

#1) Spin up a new postgres database



#2) Speed

Basic setup

```
>>> make build  
>>> make test
```

Basic setup - make build

```
13 build:  
14     @echo "Building the tester27 and tester36 docker images"  
15     docker build . \  
16         -f tests/Dockerfile \  
17         --build-arg PYTHON_VERSION=2.7 \  
18         -t tester27  
19     docker build . \  
20         -f tests/Dockerfile \  
21         --build-arg PYTHON_VERSION=3.6 \  
22         -t tester36
```

Basic setup - make build

```
1 ARG PYTHON_VERSION
2 FROM python:$PYTHON_VERSION
3
4 LABEL maintainer=zmarine@squarespace.com
5 VOLUME /opt
6 WORKDIR /opt
7
8 COPY requirements.txt /tmp
9 COPY tests/test-requirements.txt /tmp
10 RUN pip install --no-cache-dir -r /tmp/test-requirements.txt -r /tmp/requirements.txt
11
12 CMD ["python", "-m", "pytest", "/opt/tests"]
```

Basic setup - make build

```
1 ARG PYTHON_VERSION
2 FROM python:$PYTHON_VERSION
3
4 LABEL maintainer=zmarine@squarespace.com
5 VOLUME /opt
6 WORKDIR /opt
7
8 COPY requirements.txt /tmp
9 COPY tests/test-requirements.txt /tmp
10 RUN pip install --no-cache-dir -r /tmp/test-requirements.txt -r /tmp/requirements.txt
11
12 CMD ["python", "-m", "pytest", "/opt/tests"]
```

Using Python 2 (or 3)

Basic setup - make build

```
1 ARG PYTHON_VERSION
2 FROM python:$PYTHON_VERSION
3
4 LABEL maintainer=zmarine@squarespace.com
5 VOLUME /opt
6 WORKDIR /opt
7
8 COPY requirements.txt /tmp
9 COPY tests/test-requirements.txt /tmp
10 RUN pip install --no-cache-dir -r /tmp/test-requirements.txt -r /tmp/requirements.txt
11
12 CMD ["python", "-m", "pytest", "/opt/tests"]
```

Using Python 2 (or 3)

Install our requirements

Basic setup - make build

```
1 ARG PYTHON_VERSION
2 FROM python:$PYTHON_VERSION
3
4 LABEL maintainer=zmarine@squarespace.com
5 VOLUME /opt
6 WORKDIR /opt
7
8 COPY requirements.txt /tmp
9 COPY tests/test-requirements.txt /tmp
10 RUN pip install --no-cache-dir -r /tmp/test-requirements.txt -r /tmp/requirements.txt
11
12 CMD ["python", "-m", "pytest", "/opt/tests"]
```

Using Python 2 (or 3)

Install our requirements

And run pytest

⌚ Basic setup - make test

```
74 test: clean create_network start_postgres sleep test27 test36 stop_postgres
```

Additional Testing Nuances

- Want each test to be independent
- Tests require setup

Additional Testing Nuances

- Want each test to be independent → Roll back changes after each test
- Tests require setup → Use pytest fixtures for setup

Additional Testing Nuances

Yield fixtures separates test setup and test teardown:

```
29 @pytest.fixture(scope='function')
30 def cursor():
31     host = 'pgbedrock_postgres' if os.environ.get('WITHIN_DOCKER_FLAG') else 'localhost'
32     db_config = {'host': host,
33                  'port': 5432,
34                  'user': 'test_user',
35                  'password': 'test_password',
36                  'dbname': 'test_db'}
37     db_connection = psycopg2.connect(**db_config)
38     cursor = db_connection.cursor(cursor_factory=psycopg2.extras.DictCursor)
39     yield cursor
40     db_connection.rollback()
41     db_connection.close()
```

Additional Testing Nuances

Yield fixtures separates test setup and test teardown:

```
29 @pytest.fixture(scope='function')
30 def cursor():
31     host = 'pgbedrock_postgres' if os.environ.get('WITHIN_DOCKER_FLAG') else 'localhost'
32     db_config = {'host': host,
33                  'port': 5432,
34                  'user': 'test_user',
35                  'password': 'test_password',
36                  'dbname': 'test_db'}
37     db_connection = psycopg2.connect(**db_config)
38     cursor = db_connection.cursor(cursor_factory=psycopg2.extras.DictCursor)
39     yield cursor
40     db_connection.rollback()
41     db_connection.close()
```

Setup ↑

↓ Teardown

Additional Testing Nuances

Yield fixtures separates test setup and test teardown:

```
29 @pytest.fixture(scope='function')
30 def cursor():
31     host = 'pgbedrock_postgres' if os.environ.get('WITHIN_DOCKER_FLAG') else 'localhost'
32     db_config = {'host': host,
33                  'port': 5432,
34                  'user': 'test_user',
35                  'password': 'test_password',
36                  'dbname': 'test_db'}
37     db_connection = psycopg2.connect(**db_config)
38     cursor = db_connection.cursor(cursor_factory=psycopg2.extras.DictCursor)
39     yield cursor
40     db_connection.rollback()
41     db_connection.close()
```

Setup ↑

↓ Teardown

Additional Testing Nuances

Yield fixtures separates test setup and test teardown:

```
29 @pytest.fixture(scope='function')
30 def cursor():
31     host = 'pgbedrock_postgres' if os.environ.get('WITHIN_DOCKER_FLAG') else 'localhost'
32     db_config = {'host': host,
33                  'port': 5432,
34                  'user': 'test_user',
35                  'password': 'test_password',
36                  'dbname': 'test_db'}
37     db_connection = psycopg2.connect(**db_config)
38     cursor = db_connection.cursor(cursor_factory=psycopg2.extras.DictCursor)
39     yield cursor
40     db_connection.rollback()
41     db_connection.close()
```

Setup ↑

↓ Teardown

Agenda

1. Config As Code
2. Ansible
3. Testing

Questions?

Zach Marine, Squarespace



zmarine@squarespace.com



[zcmarine](https://github.com/zcmarine)



[@zcmarine](https://twitter.com/zcmarine)