25. hroug

godišnja konferencija

Pythian

Patching with Ansible

What worked, What didn’t work, and Why

Timur Akhmadeev
HrOUG 2021

About Pythian

24
Years in Business

400+
Experts Across 5 Continents

500+
Customers Globally

L❤VE YOUR DATA

About Pythian

24
Years in Business

400+
Experts Across 5 Continents

500+
Customers Globally

Google Cloud

Google Cloud GLOBAL Partner of the Year Data Management 2020

Google Cloud Partner Managed Service Provider

Premier Partner

140+ Certifications

8 Specializations

amazon web services

aws 50 CERTIFIED AWS PARTNER NETWORK

Advanced Partner

175+ Certifications

Azure

Gold Partner

15+ Certifications

ORACLE

Platinum Partner

150+ Certifications

SAP

SAP Certified Partner

40+ Certifications

About Me

Dev => Perf => DBA => Apps DBA

About Me

Dev => Perf => DBA => Apps DBA

16+ years with the Database and Java

About Me

Dev => Perf => DBA => Apps DBA

16+ years with the Database and Java

Systems Performance and Architecture

About Me

Dev => Perf => DBA => Apps DBA

16+ years with the Database and Java

Systems Performance and Architecture

https://timurakhmadeev.wordpress.com

https://pythian.com/blog/author/akhmadeev

https://twitter.com/tmmdv

timur.akhmadeev@gmail.com

<ad>

</ad>

<ad>

Planning a trip to Moscow?

</ad>

<ad>

Planning a trip to Moscow?
Have a presentation to share?

</ad>

<ad>

Planning a trip to Moscow?
Have a presentation to share?

Email me timur.akhmadeev@gmail.com

We’ll organize a meetup!

</ad>

Agenda

background

initial state

first steps

missing bits

ansible

issues

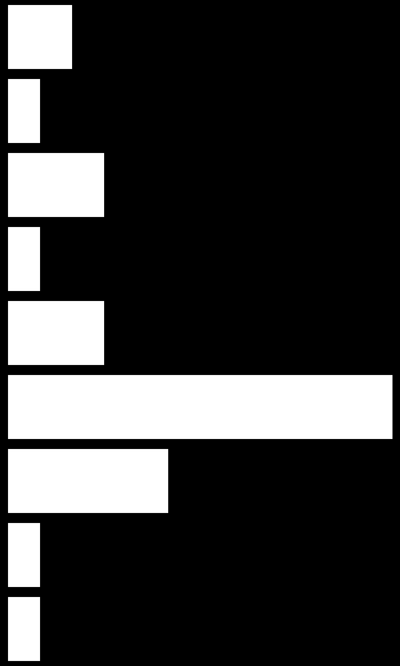
results

plans

summary

Agenda %

background
initial state
first steps
missing bits
ansible
issues
results
plans
summary



background

Pythian

- DEV, TEST, PROD
- ~15 services per environment

- DEV, TEST, PROD
- ~15 services per environment
- DBs + FMW 11g/12c + eBS running in two ODA VP
- regular security patching 4 times a year

- DEV, TEST, PROD
- ~15 services per environment
- DBs + FMW 11g/12c + eBS running in two ODA VP
- regular security patching 4 times a year
- 50-70 patches per cycle
- additional maintenance

initial state

Pythian

Basics

- on most servers
  - start/stop scripts
  - backup script
- partially templated documentation

Automation discussions

- ~known efforts per patch cycle with existing approach

Automation discussions

- ~known efforts per patch cycle with existing approach
- estimated time to get benefits from automation: 2+ years

Too much copy-paste

- copy pwd from the secret store
- paste pwd into putty session

Too much copy-paste

- copy pwd from the secret store
- paste pwd into putty session
- for each group of commands:
    - copy & paste commands into shell
    - wait for the output
    - analyze & copy-paste output for reference

Too much copy-paste

- copy pwd from the secret store
- paste pwd into putty session
- for each group of commands:
    - copy & paste commands into shell
    - wait for the output
    - analyze & copy-paste output for reference
- rinse and repeat

first steps

Pythian

do-this.sh

- configured password-less connectivity to most hosts

do-this.sh

- configured password-less connectivity to most hosts
- `./do-this.sh servers.conf "command"`
- 100 lines of bash

do-this.sh

- configured password-less connectivity to most hosts
- `./do-this.sh servers.conf "command"`
- 100 lines of bash
- parallel calls with waits for specific stages
- summary report with return codes and timing

servers.conf

```
#parallel=4
oracle:server1
oracle:server2
#wait
oracle:server3
#wait
oracle:server4
user5:server5
user6:server6
user7:server7
```

```
echo "Summary results of executing $CMD on all $SERVERS servers"
line='................................'
for x in "${!results[@]}"; do
  printf "%s %s %s" $x "${line:${#x}}" "${results[$x]}"
  printf "... %sm:%ss\n" $((${tim[$x]}/60)) $((${tim[$x]}%60))
done
printf "Total time: %sm:%ss\n" $((${t_total}/60))
$((${t_total}%60))
```

```
Summary results of executing ~/stop.sh on all servers.conf servers
oracle:server-one1 ............. 0 ... 5m:32s
oracle:server-two .............. 0 ... 1m:37s
...
oracle:server3 ................. 0 ... 1m:54s
user1:server4 .................. 0 ... 0m:16s
oracle:server5 ............... 127 ... 1m:54s
Total time: 26m:47s
```

```
backup_fmw_[dev/test/prod].sh
stop_fmw_[dev/test/prod].sh
check_fmw_[dev/test/prod].sh
start_fmw_[dev/test/prod].sh
sanity_check_[dev/test/prod].sh
```

missing bits

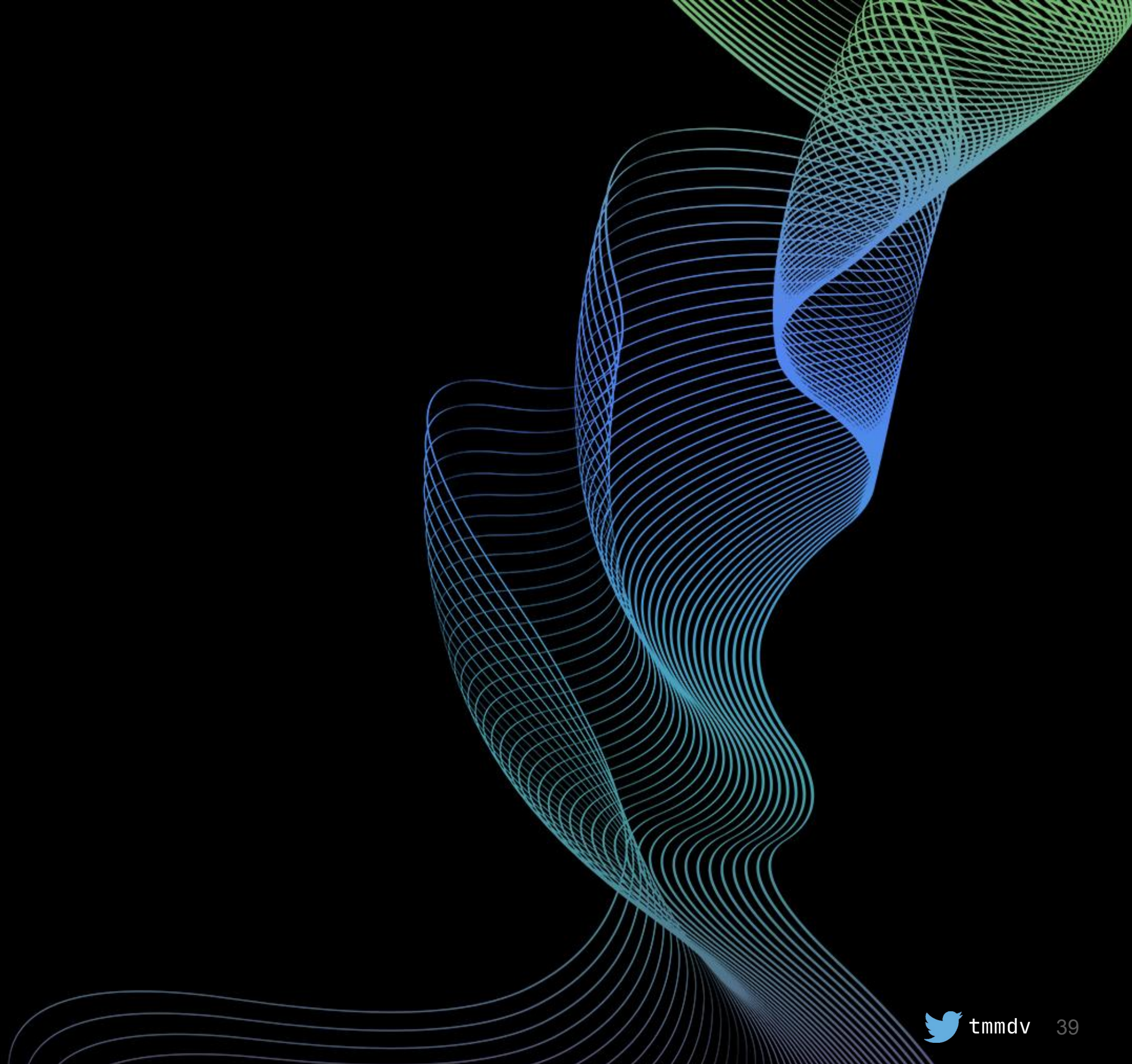
- eBS stop/start
    - store & pass credentials

- eBS stop/start
  - store & pass credentials
- patching
  - seemed way too complex for pure bash

'twas 2020

'twas 2020, 'twas boring

ansible



Why ansible?

- agentless
- relatively easy install
  - most Linux distributions have it in standard repos
- passwordless auth already in place for most servers

Expectations

- simple
- less manual work
- faster patching
- more robust

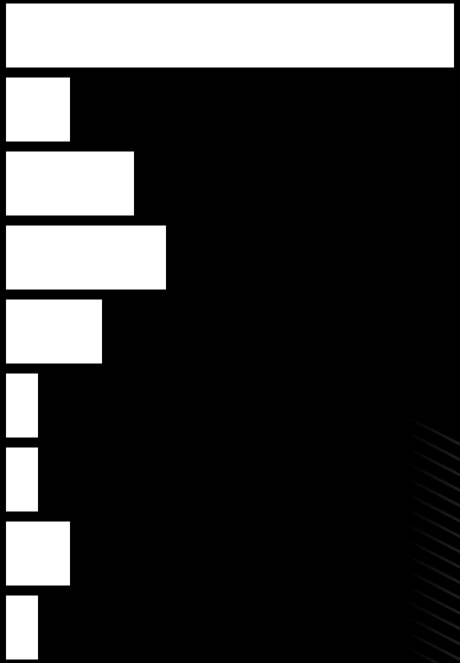
Automations

- patch download & staging
- backups
- shutdown
- WLS 11g patching
- WLS 12c patching
- JDK patching
- checking JDK symlinks
- startup
- sanity checking

Automations

- patch download & staging
- backups
- shutdown
- WLS 11g patching
- WLS 12c patching
- JDK patching
- checking JDK symlinks
- startup
- sanity checking

Efforts



No automations

- parts of FMW 11g patching (draft in place)

No automations

- parts of FMW 11g patching (draft in place)
- eBS patching
  - 1 or 2 instances per cycle
  - relatively complex
  - most things done online prior to maintenance window

No automations

- parts of FMW 11g patching (draft in place)
- eBS patching
  - 1 or 2 instances per cycle
  - relatively complex
  - most things done online prior to maintenance window
- DB homes patching
  - small portion of time spent

No automations

- parts of FMW 11g patching (draft in place)
- eBS patching
  - 1 or 2 instances per cycle
  - relatively complex
  - most things done online prior to maintenance window
- DB homes patching
  - small portion of time spent
- FMW upgrades
  - rare, complex, manual

issues

fresh ansible version is required

fresh ansible version is required

>= 2.9

need an up-to-date `jinja2` too

testing in a local dbg env

testing in a local dbg env
WSL => Ubuntu

testing in a local dbg env
WSL => Ubuntu
isn't same as testing in DEV

time to write simple stuff?

time to write simple stuff? *minutes*

time to write simple stuff? *minutes*
time to write a loop?

time to write simple stuff? *minutes*
time to write a loop? *a week*

```
patches:
  - patch_type: wls10
    platform: 226P
    patch_list:
      - patch_number: "31178492"
        patch_desc: "Patch 31178492: WLS PATCH SET UPDATE 10.3.6.0.200714"
        patch_file: p31178492_1036_Generic.zip
      - patch_number: "13845626"
        patch_desc: "SU Patch [DTN2]: 10.3.6.0.200714WLSPSU Overlay: ..."
        patch_file: p13845626_10360200714_Generic.zip
      ...
  - patch_type: webtier
    platform: 226P
    patch_list:
      - patch_number: "31304503"
        patch_desc: "OSS BUNDLE PATCH 11.1.1.9.200714"
        patch_file: p31304503_111190_Linux-x86-64.zip
        uncompress: true
      ...
```

```
- name: check already downloaded patches
  stat:
    path: "{{local_stage_dir}}/{{item.0.patch_type}}/{{item.1.patch_file}}"
  loop: "{{ patches | subelements('patch_list') }}"
  register: existing_patches
```

```
- name: download patches
  shell: |
    java -jar getMOSPatch.jar patch={{ item.item.1.patch_number }} \
      platform={% if item.item.1.platform is defined %}{{
item.item.1.platform }}{% else %}{{ item.item.0.platform }}{% endif %} \
      regexp=.*{{ item.item.1.patch_file | replace('.zip','') }}.* \
      stagedir={{ local_stage_dir }}/{{ item.item.0.patch_type }}
download=all \
      MOSUser={{ MOSUser }} MOSPass={{ MOSPass }} silent=yes debug=yes
  ignore_errors: yes
  when: item.stat.exists == False
  loop: "{{ existing_patches.results }}"
```

```
- name: download patches
  shell: |
    java -jar getMOSPatch.jar patch={{ item.item.1.patch_number }} \
      platform={% if item.item.1.platform is defined %}{{
item.item.1.platform }}{% else %}{{ item.item.0.platform }}{% endif %} \
      regexp=.*{{ item.item.1.patch_file | replace('.zip','') }}.* \
      stagedir={{ local_stage_dir }}/{{ item.item.0.patch_type }}
download=all \
      MOSUser={{ MOSUser }} MOSPass={{ MOSPass }} silent=yes debug=yes
  ignore_errors: yes
  when: item.stat.exists == False
  loop: "{{ existing_patches.results }}"
```

logging

logging is poor

deal with a long cryptic out

deal with a long cryptic out

or

nothing

need a balanced combination: compact terminal output + detailed log in a file

detailed log can expose secrets

```yaml
- name: stop ebs
  become: true
  become_user: "{{ oracle_user }}"
  ignore_errors: yes
  no_log: "{{ no_debug | default(true) }}"
  shell: |
    ...
```

```yaml
- hosts: some-hosts
  # can't be variable: https://github.com/ansible/ansible/issues/18131
  serial: 2
  tasks:
    ...
```

`serial` with two inventories

`serial` with two inventories works as if there's one large inventory

```
- hosts:
    - group1
    - group2
  serial:
    - 1
    - 2
  tasks:
    - name: start those in a specific order
      shell: "..."
```

```
inv/dev:

group1:
  server1
group2:
  server2
  server3
```

```
inv/dev:

group1:
  server1
group2:
  server2
  server3
```

```
inv/test:

group1:
  server1
group2:
  server2
  server3
```

Pythian

```
inv/dev:                inv/test:

group1:                 group1:
  server1                 server1
group2:                 group2:
  server2                 server2
  server3                 server3
```

```
batch 1: dev.server1
batch 2: test.server1, dev.server2
batch 3: dev.server3, test.server2
batch 4: test.server3
```

```
ansible-playbook -i inv/dev -i inv/test start.yml
```

```
~~ansible-playbook -i inv/dev -i inv/test start.yml~~
```

```
ansible-playbook -i inv/dev  start.yml
ansible-playbook -i inv/test start.yml
```

`serial` with profile_tasks

`serial` with profile_tasks does not work properly

https://github.com/ansible-collections/ansible.posix/issues/83

```
$ grep profile ansible.cfg
callback_whitelist = profile_tasks

$ cat inv.yml
all:
  hosts:
    host1:
      ansible_host: localhost
    host2:
      ansible_host: localhost

$ cat test.yml
- hosts: all
  connection: local
  serial: 1
  tasks:
    - name: sleep 3s
      shell: sleep 3
```

```
sleep 3s ---------------------- 3.45s

real  0m7.907s
user  0m1.664s
sys   0m0.507s
```

```
$ cat test.yml
- hosts: localhost
  connection: local
  gather_facts: no
  vars:
    var1: true
  tasks:
    - name: check var1 value
      shell: echo '{{var1}}'
```

```
changed: [localhost] => {
  "changed": true,
  "cmd": "echo 'True'",
  "delta": "0:00:00.015293",
  "end": "2021-08-01 21:19:16.262756",
  "rc": 0,
  "start": "2021-08-01 21:19:16.247463"
}
```

```
$ ansible-playbook -e var1=true test.yml -v
$ ansible-playbook -e var1=true test.yml -v

PLAY [localhost] ************************************************
TASK [check var1 value] *****************************************
changed: [localhost] => {
  "changed": true,
  "cmd": "echo 'true'",
  "delta": "0:00:00.015678",
  "end": "2021-08-02 21:13:03.056332",
  "rc": 0,
  "start": "2021-08-02 21:13:03.040654"
}
```

```
$ cat test.yml
- hosts: localhost
  connection: local
  gather_facts: no
  vars:
    var1:
      x: "this is x"
      y: "this is y"
  tasks:
    - name: do something with var1
      shell: echo '{{var1.x}}', '{{var1.y}}'
```

```
$ ansible-playbook -e '{"var1":{"x":"this is x2", "y":"this is y"}}'
test.yml -v
...
changed: [localhost] => {
    "changed": true,
    "cmd": "echo 'this is x2', 'this is y'",
    "delta": "0:00:00.015026",
    "end": "2021-08-03 21:43:38.203561",
    "rc": 0,
    "start": "2021-08-03 21:43:38.188535"
}
...
```

```
$ tree -a
.
├── group_vars
│   └── fmw
└── inv
    ├── dev
    │   ├── dev
    │   └── group_vars
    └── test
        ├── group_vars
        └── test

$ cat group_vars/fmw
remote_stage_top: /u01/patches
```

Pythian

```
$ tree -a
.
├── group_vars
│   └── fmw
└── inv
    ├── dev
    │   ├── dev
    │   └── group_vars
    └── test
        ├── group_vars
        └── test

$ cat group_vars/fmw
remote_stage_top: /u01/patches
```

```
$ cat inv/dev/dev
all:
  children:
    fmw:
      hosts:
        host1-dev:
          ansible_host: localhost

$ cat inv/test/test
all:
  children:
    fmw:
      hosts:
        host1-test:
          ansible_host: localhost
```

Pythian

```
$ ansible -i inv/dev -i inv/test -m debug -a "msg={{remote_stage_top}}" fmw
$ ansible -i inv/dev -i inv/test -m debug -a "msg={{remote_stage_top}}" fmw
host1-dev | SUCCESS => {
   "msg": "/u01/patches"
}
host1-test | SUCCESS => {
   "msg": "/u01/patches"
}
```

```
$ echo "remote_stage_top: /home/oracle" > inv/dev/group_vars/fmw
$ echo "remote_stage_top: /home/oracle" > inv/dev/group_vars/fmw

$ ansible -i inv/dev -i inv/test -m debug -a "msg={{remote_stage_top}}" fmw
host1-dev | SUCCESS => {
   "msg": "/u01/patches"
}
host1-test | SUCCESS => {
   "msg": "/u01/patches"
}
```

```
$ mkdir inv/dev/host_vars
$ echo "remote_stage_top: /home/oracle" > inv/dev/host_vars/host1-dev
$ mkdir inv/dev/host_vars
$ echo "remote_stage_top: /home/oracle" > inv/dev/host_vars/host1-dev


$ ansible -i inv/dev -i inv/test -m debug -a "msg={{remote_stage_top}}" fmw
host1-dev | SUCCESS => {
   "msg": "/home/oracle"
}
host1-test | SUCCESS => {
   "msg": "/u01/patches"
}
```

```
$ cat test.yml
- hosts: host1
  gather_facts: no
  tasks:
    - name: copy and unzip a file
      unarchive:
        src: p32218454_190000_Linux-x86-64.zip
        dest: /home/tiak/
```

```
Saturday 07 August 2021  21:50:13 +0300 (0:03:29.166) 0:03:29.308
=======================================================================
copy and unzip a file ----------------------------------------- 209.17s
```

Pythian

```
$ time scp -P10022 ~/p32218454_190000_Linux-x86-64.zip
tiak@127.0.0.1:/home/tiak/
p32218454_190000_Linux-x86-64.zip       100% 1426MB  53.6MB/s   00:26

real    0m27.526s
user    0m9.188s
sys     0m6.781s

$ time unzip -qo p32218454_190000_Linux-x86-64.zip

real    0m54.259s
user    0m28.821s
sys     0m7.505s
```

using `copy` module then calling `unzip` via shell instead of `unarchive`

other issues: RAM usage and permissions

remote_tmp

`remote_tmp`

defaults to

`$HOME/.ansible/tmp`

I've patched wls10 in DEV

I've patched wls10 in DEV
when we were patching TEST

```yaml
- hosts: fmw
  vars_prompt:
    - name: current_env
      prompt: please confirm current environment
      private: no
  tasks:
    - name: check current environment
      fail:
        msg: '{{__env_name}}' is different from '{{current_env}}'
      when: current_env != __env_name
```

Pythian

```
$ ansible-playbook -i inv/test patch-jdk.yml -v
$ ansible-playbook -i inv/test patch-jdk.yml -v
Using $PATH/ansible.cfg as config file
please confirm current environment:
```

patch cycle preparations

patch cycle preparations are tricky

patch cycle preparations
are tricky
and manual

manual steps can result in error

manual steps can result in error due to human error

applied wrong patch
due to copy-paste error

results

Pythian

- first drafts: about 2 weeks of efforts

- first drafts: about 2 weeks of efforts
- first patch cycle: some things worked
- bugs noted & fixed in the following weeks

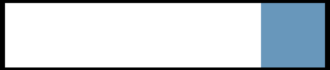
- first drafts: about 2 weeks of efforts
- first patch cycle: some things worked
- bugs noted & fixed in the following weeks
- next few cycles - same thing, but less bugs

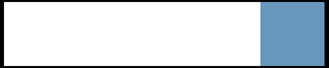
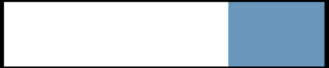
- first drafts: about 2 weeks of efforts
- first patch cycle: some things worked
- bugs noted & fixed in the following weeks
- next few cycles - same thing, but less bugs
- playbooks change based on learnings
- efforts to make code stable: ~1 month

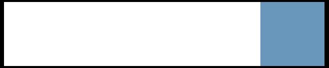
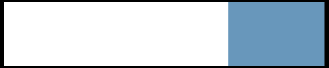
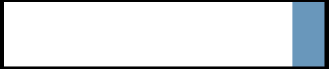
- normally we have very few issues with dev run now
- almost no issues with test and prod

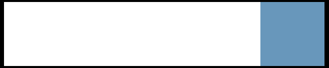
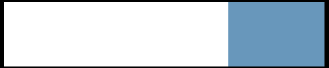
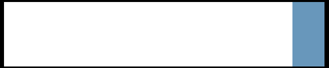
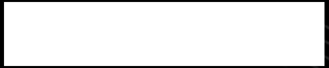
Expectations	Reality

Pythian

Expectations	Reality
simple	

Expectations	Reality
simple	
less manual work	

Expectations	Reality
simple	
less manual work	
faster patching	

Expectations	Reality
simple	
less manual work	
faster patching	
more robust	

plans

Pythian

- simplify patch cycle preparations

- simplify patch cycle preparations
- add database patching after 19c upgrades

- simplify patch cycle preparations
- add database patching after 19c upgrades
- re-work

summary

- patching can benefit from ansible
- the more you repeat playbooks, the more benefits you get

- patching can benefit from ansible
- the more you repeat playbooks, the more benefits you get
- relatively simple to start work with
- takes some time to develop

- patching can benefit from ansible
- the more you repeat playbooks, the more benefits you get
- relatively simple to start work with
- takes some time to develop
- solves a few issues
- adds other issues

- patching can benefit from ansible
- the more you repeat playbooks, the more benefits you get
- relatively simple to start work with
- takes some time to develop
- solves a few issues
- adds other issues
- the more you use automation, the more automation you want
- reasonable to automate most repetitive tasks

Tim MalcomVetter @malcomvetter

OH:

A: "Just use Terrible to deploy it."

B: "What?"

A: "Terraform and Ansible."

B: "Oh. Yeah. Terrible."

7:17 PM · Nov 20, 2020·Twitter Web App

Thank you!

Feedback is welcome:

timur.akhmadeev@gmail.com

Pythian

q&a