No Compromise: Use Ansible properly or stick to your scripts

Björn Meier

BlueYonder
Best decisions, delivered daily

Short Ansible Example

inventory

```
[webserver]
host1

[database]
host2

[my_app:children]
webserver
database
```

playbook

```yaml
- hosts: webserver
  tasks:
    - apt:
        name: "{{ item }}"
        state: present
      with_items:
        - nginx
        - python3

- hosts: database
  tasks:
    - apt:
        name: "{{ item }}"
        state: present
      with_items:
        - postgresql
        - postgresql-client
```

```
514 ↪  ansible-playbook -i hosts playbook.yml

PLAY [webserver] ***************************************************************

TASK [setup] *******************************************************************
Tuesday 24 October 2017  22:02:44 +0200 (0:00:00.104)       0:00:00.104 *******
ok: [host1]

TASK [apt] *********************************************************************
Tuesday 24 October 2017  22:02:45 +0200 (0:00:01.654)       0:00:01.759 *******
changed: [host1] => (item=[u'nginx', u’python3'])

PLAY [database] ****************************************************************

TASK [setup] *******************************************************************
Tuesday 24 October 2017  22:02:47 +0200 (0:00:01.623)       0:00:03.382 *******
ok: [host2]

TASK [apt] *********************************************************************
Tuesday 24 October 2017  22:02:49 +0200 (0:00:01.815)       0:00:05.198 *******
changed: [host2] => (item=[u'postgresql', u’postgresql-client'])

PLAY RECAP *********************************************************************
host1                      : ok=2    changed=1    unreachable=0    failed=0
host2                      : ok=2    changed=1    unreachable=0    failed=0
```

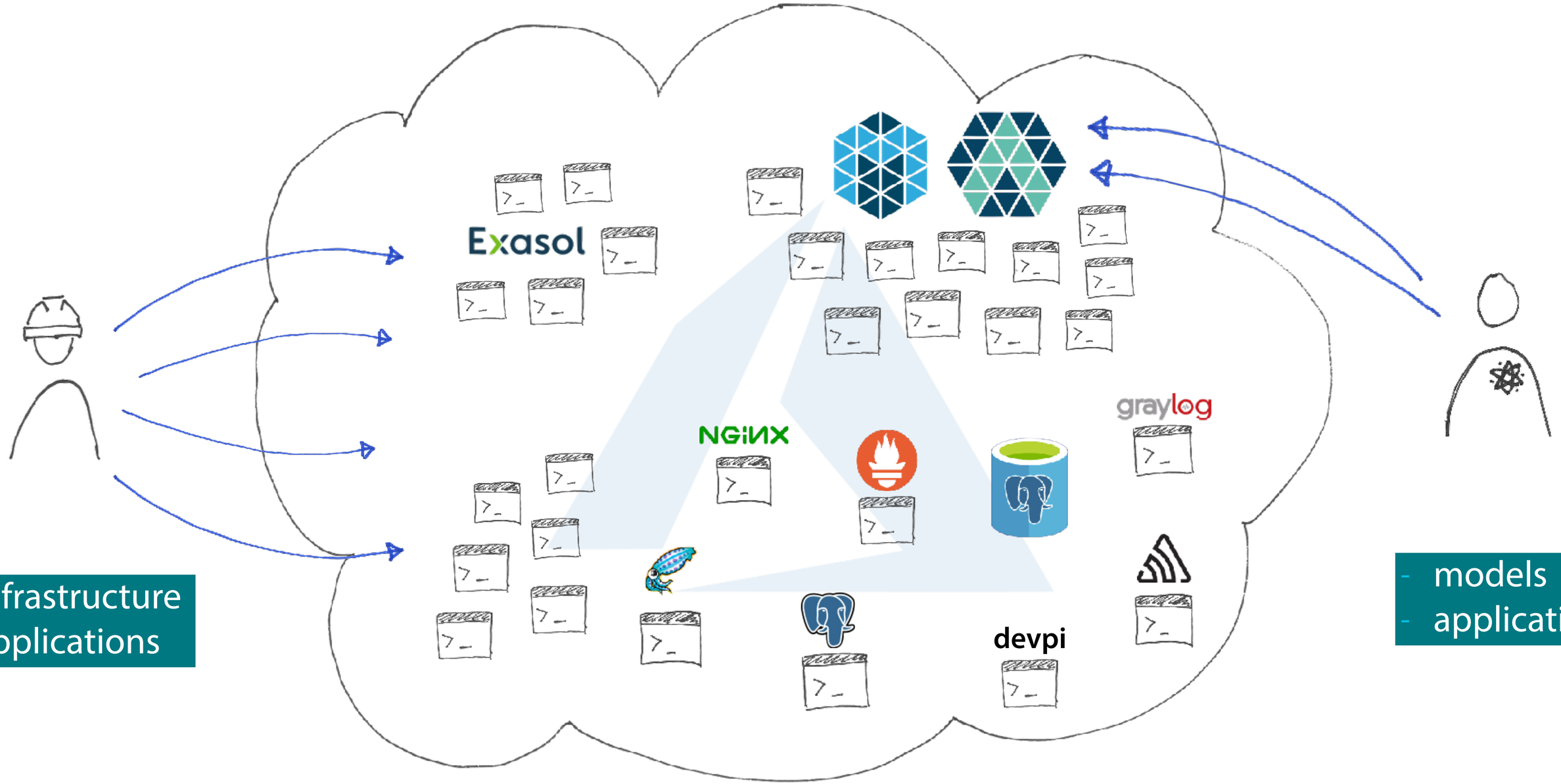
--check mode

- no change is executed
- changes are simulated

--diff mode

changes are shown

Where do we use Ansible



- infrastructure
- applications

- models
- applications

How we started

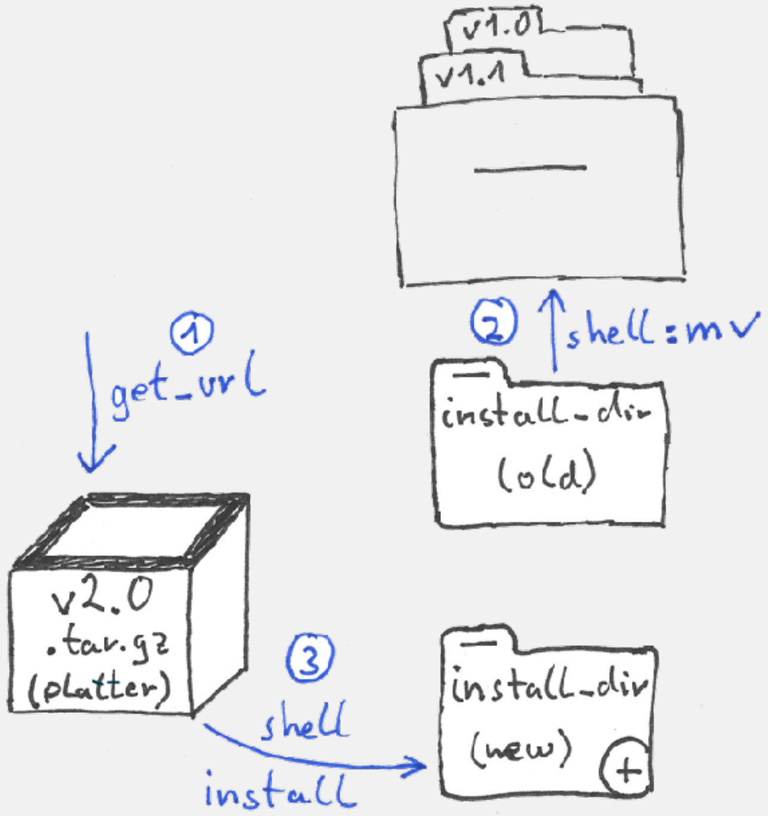
bundled applications

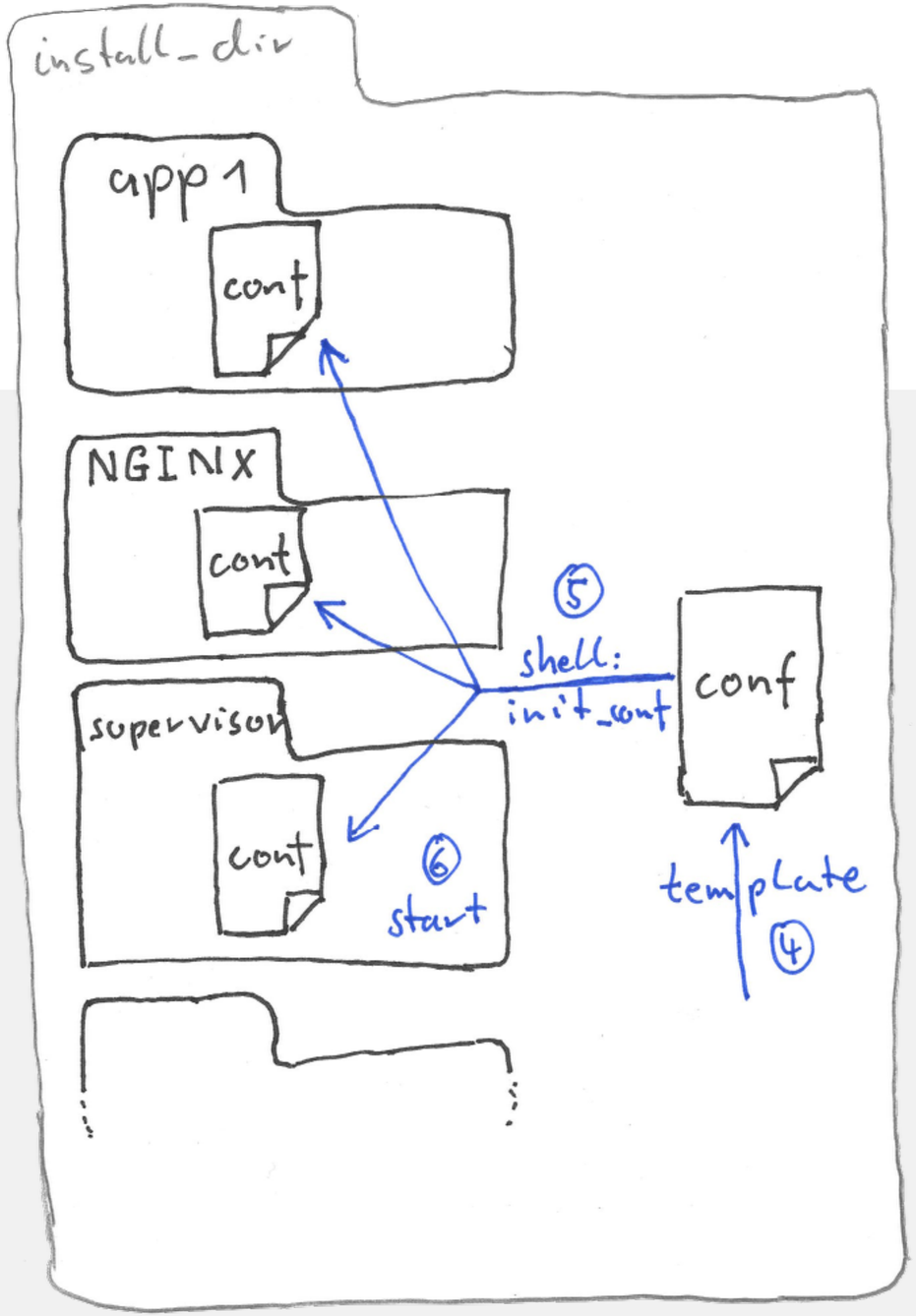
- python applications
- NGINX
- supervisor
- …

manual deployment mapped to Ansible tasks

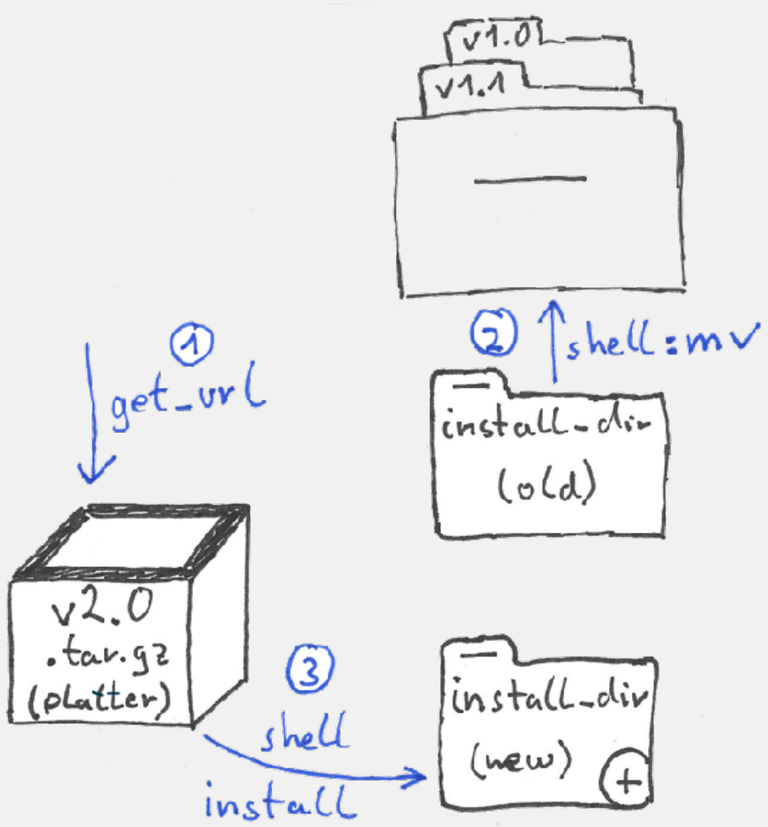
- prepare hosts
- deploy bundled applications

How we started



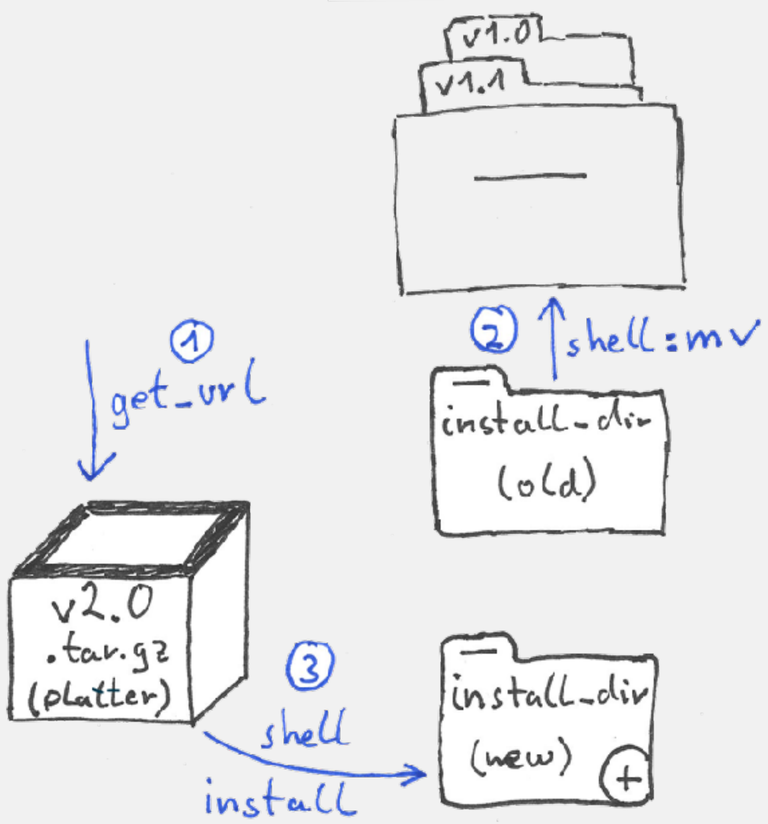


Application Deployment Problems




```yaml
vars:
  version: "v2.0"

tasks:
  - get_url:
      url: "http://storage.local/app-{{ version }}.tar.gz"
      dest: /app

  - shell: "mv /app/install /app/old"

  - unarchive:
      src: "/app/app-{{ version }}.tar.gz"
      dest: /app

  - shell: "./install /app/install"
```

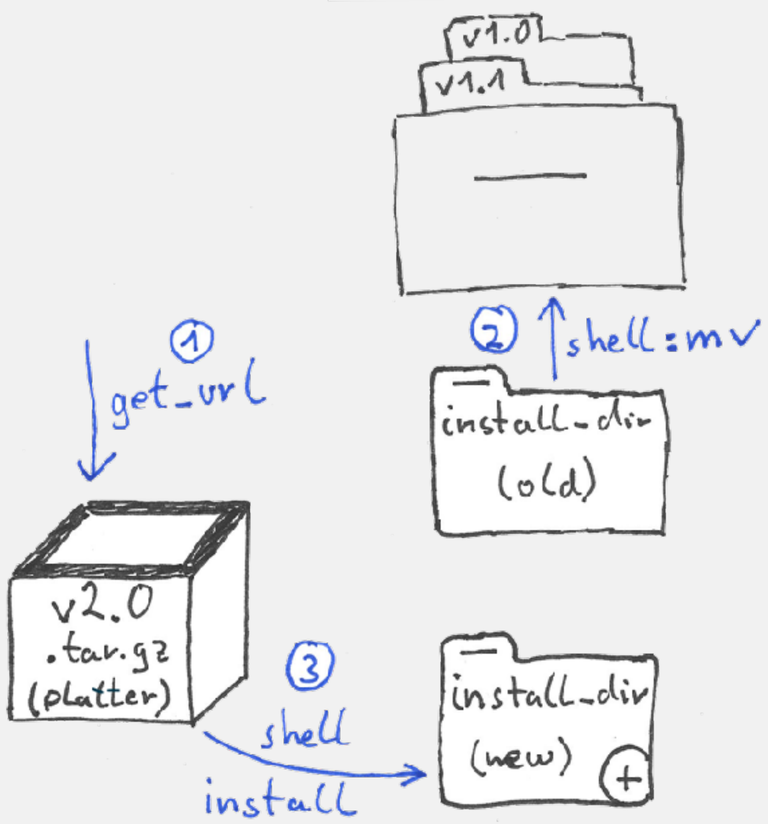
Application Deployment Problems



always changed

```
PLAY RECAP *********************************************************
host    : ok=37   changed=12    unreachable=0    failed=0
```

Application Deployment Problems

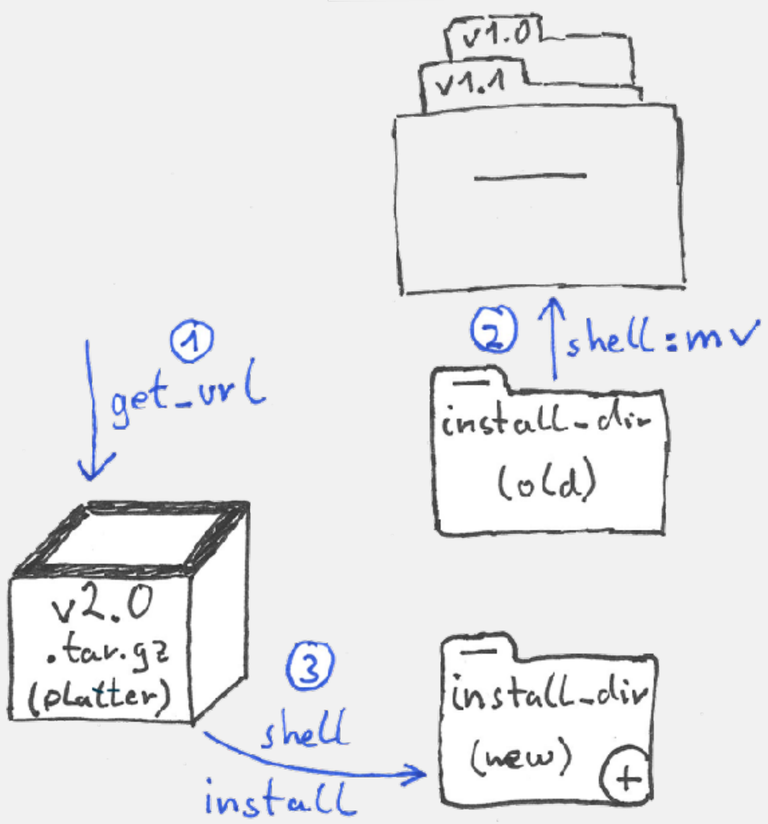


skipped in check mode

```
TASK [command] *****************************************
Monday 23 October 2017  11:07:37 +0200 (0:00:01.360) *
skipping: [host]
```

```
TASK [debug] *******************************************
Monday 23 October 2017  11:07:37 +0200 (0:00:00.439) *
ok: [host] => {
    "result": {
        "changed": false,
        "msg": "remote module (command) does not
support check mode",
        "skipped": true
    }
}
```

Application Deployment Problems



old

```yaml
vars:
  version: "v2.0"

tasks:
  - get_url:
      url: "http://storage.local/app-{{ version }}.tar.gz"
      dest: /app

  - shell: "rm /app/old; mv /app/install /app/old"

  - unarchive:
      src: "/app/app-{{ version }}.tar.gz"
      dest: /app

  - shell: "./install /app/install"
```

new

```yaml
tasks:
  - bundle_deploy:
      version: "{{ version }}"
      host_url: "http://storage.local/"
      install_dir: "/app/install"
```

Why an Ansible Module?

control what happens in check-mode

- would something change
- return values with changes

clean reusability inside roles/plays

- one single task
- input parameters instead of ansible variables
- direkt single result

python code

- more code flow control
- python code testing
- python libraries

A Bundle Deployment Module

```
def run_module():
    module_args = dict(version=dict(type='str', required=True),
                       host_url=dict(type='str', required=True),
                       install_dir=dict(type='str', required=True))
    result = dict(changed=False, new_version_dir='')
    module = AnsibleModule(argument_spec=module_args, supports_check_mode=True)

    result['changed'] = _version_changed(module.params['install_dir'],
                                         module.params['version'])

    if module.check_mode:
        result['new_version_dir'] = _get_install_dir(module.params['install_dir'],
                                                     module.params['version'])
        module.exit_json(**result)

    if result['changed']:
        try:
            new_version_file = _download(module.params['host_url'], module.params['version'])
            result['new_version_dir'] = _install(new_version_file, module.params['install_dir'])
            _link_new_version(module.params['install_dir'], result['new_version_dir'])
        except:
            module.fail_json(msg=sys.exc_info()[0], **result)

    module.exit_json(**result)
```

Usage/Testing

include

```
playbook.yml
.library/
    bundle_deploy.py
roles/
    my_role/
        library/
            role_module.py
```

testing

- python unit/integration tests
- playbook tests

```
- bundle_deploy:
    version: "v2.0"
    host_url: "{{ test_url }}"
    install_dir: "/tmp/module_test_dir"
  check_mode: yes
  register: check_mode_test

- assert:
    that:
      - "check_mode_test.changed"
      - “check_mode_test.new_version_director”
```

Configuration and Handler Problems

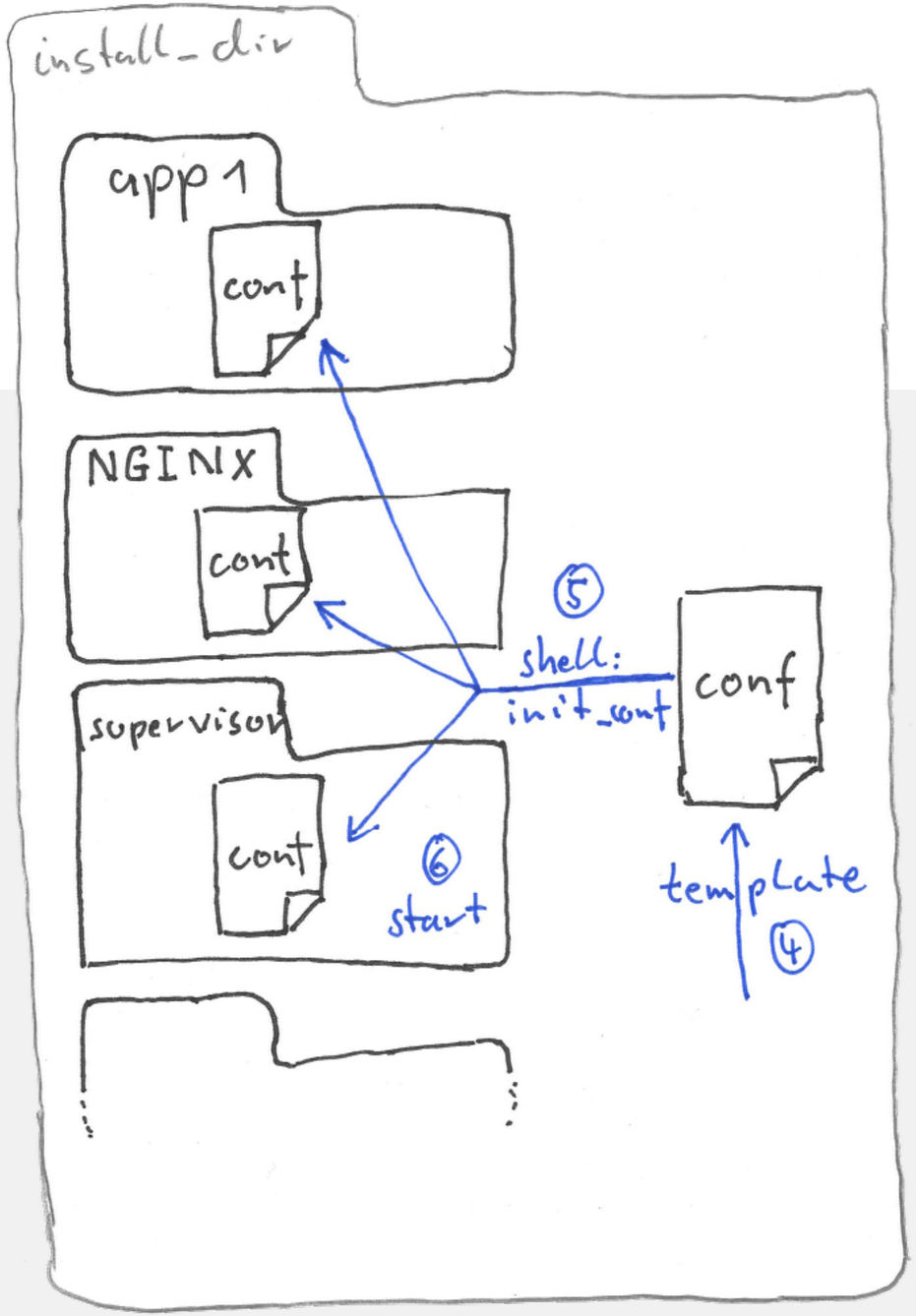
shell task to create sub configs

- always executed
- no “--diff” for all sub configs
- no “--check”

always restart all

- no change information from “init_config”
- no fine granular service restart

only main config is Ansible controlled



Configuration and Handler Problems

only one management source for

- configuration
- execution

actually use handlers

accurate handler control

- top down evaluation
- the “register” statement can be used
- notified handlers can be skipped using “when”

```
handlers:
  - name: restart group
    supervisorctl:
      name: "servicegroup:"
      state: restarted
    register: restarted_group

  - name: restart web app
    supervisorctl:
      name: "servicegroup:webapp"
      state: restarted
    when: not restarted_group is defined
```

Eat your own Dog Food

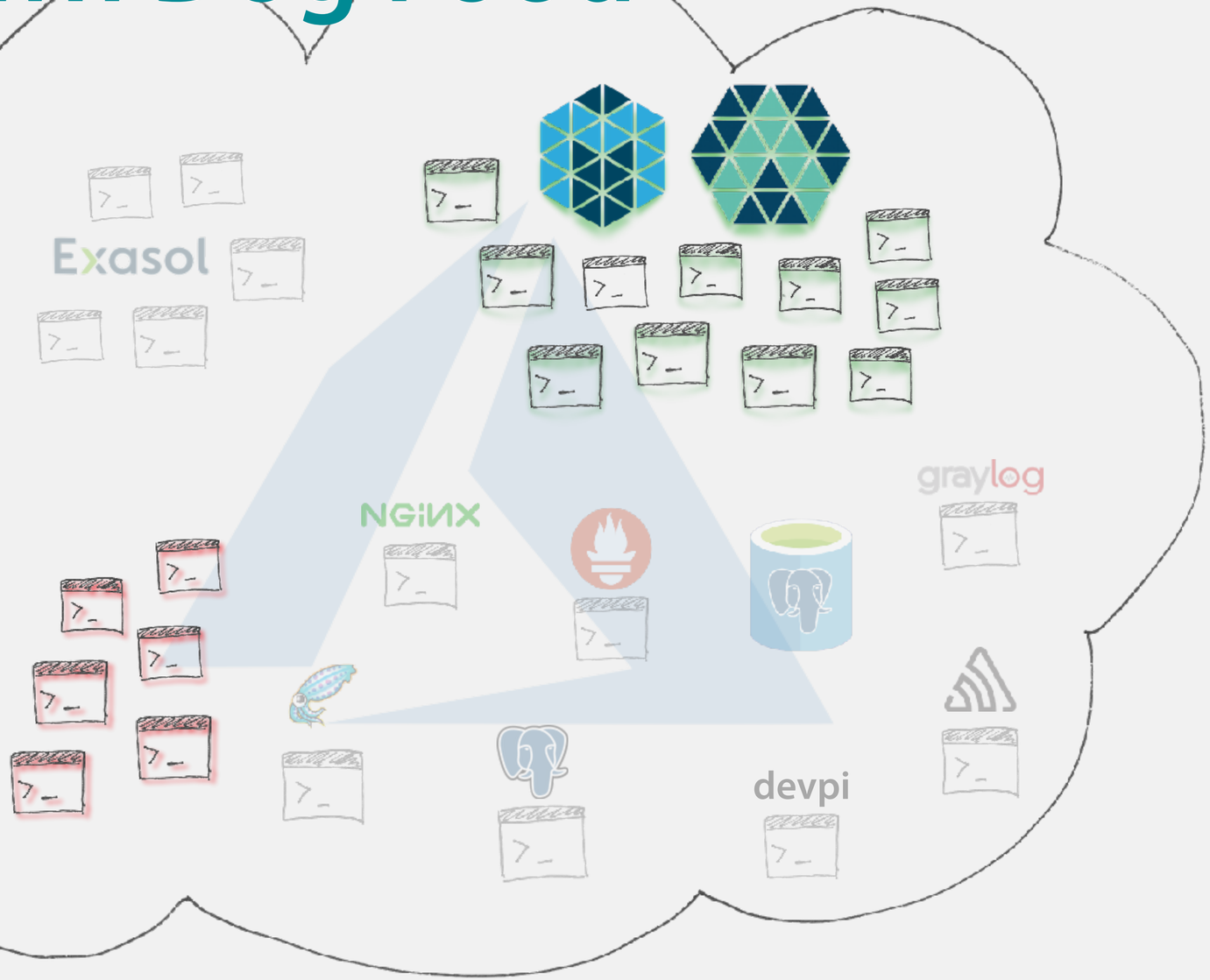
benefit from Apache Aurora with
- distributed
- redundant

running **applications**

scale and share unused resources

consume what you provide

reduce infrastructure cost and complexity



Deployment without Machines

If you can

- add
- delete
- update
- get the state of

something

-> you can manage it with Ansible

Faking it

Use hosts and groups as usual

```
[aurora-fake-ui-services]
fake-ui-services-1
fake-ui-services-1

[aurora-fake-data-services]
fake-data-services-1
fake-data-services-1

[application:children]
aurora-fake-ui-services
aurora-fake-data-services
application-databases

[aurora-fake:children]
aurora-fake-ui-services
aurora-fake-data-services
```

with fake hosts

Set group (or hosts) to local execution

`env_prod/group_vars/aurora-fake`

```
---
ansible_ssh_host: localhost
ansible_connection: local
ansible_gather_facts: no
# if necessary
ansible_python_interpreter: "/usr/bin/env python"
```

Deploy to a fake Host

```python
module = AnsibleModule(
    argument_spec=dict(
        token=dict(required=True),
        base_url=dict(required=True),
        service_identifier=dict(required=True),
        service_definition=dict(type='dict', default={})),
    supports_check_mode=True
)
# ...

if module.check_mode:
    current_status = get_service_status(module)
    result['changed'] = _compare_status(
        module.params['service_definition'],
        current_status
    )
    return result

# ...
if module.params['service_definition']['state'] == ‘present':
    create_service(module)
if module.params['service_definition']['state'] == 'absent':
    delete_service(module)
# ...
```

```yaml
- name: create and start service
  my_service_module:
    token: "{{ vault_access_token }"
    base: "{{ service_url }}"
    service_identifier: "webservice_1"
    service_definition:
      state: "started"
      requirements: "{{ requirements }}"
      instances: 4
```

Summary

- *“ansible-playbook play.yml --check --diff”* should always work
- use custom module if necessary to replace shell/command
- prevent shared responsibility between application and Ansible
- also deploy to clusters

Slides at: https://github.com/blue-yonder/documents