Continuous Integration
In challenging environments
w/
Ansible.

PyCon5 Italy, Cesare Placanica

And...

Python Milano Meetup
2016-10-19

Who am I?

- Work for Cisco Photonics Italy
- Cloud and Virtualization Group
- keobox@gmail.com


Cisco Systems

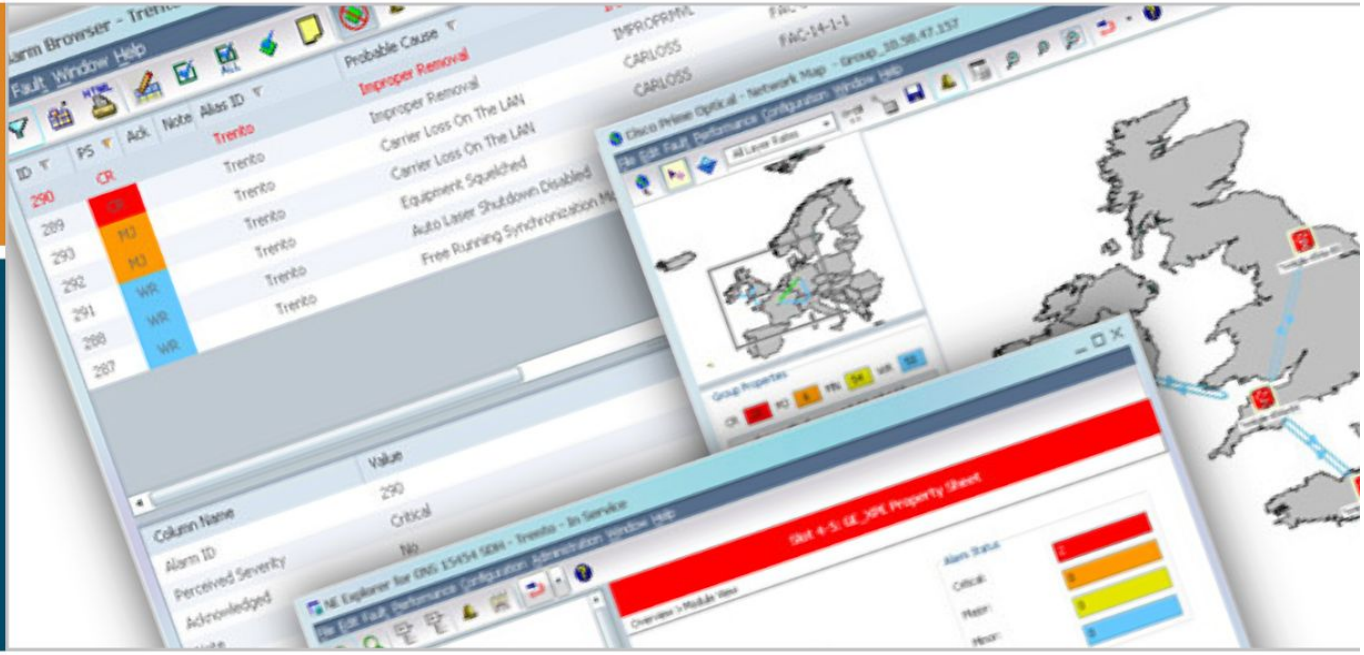

CISCO
Cisco ONS 15454
Optical Network System

lco Boosts Network Management Efficiency

work transport management platform helps Enventis centralize operations, increase availability, and control

ventis, a subsidiary of
ckoryTech (NASDAQ: HTCO)
ecommunications
nkato, MN
0

form saves time
costs
crease availability
ating assets



Cisco Prime Optical

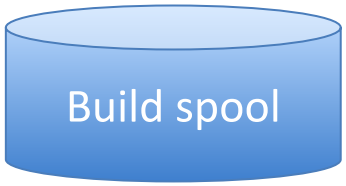
- (Mostly) Java Application.
- Server: runs on RH Linux.
- There's a GUI installer but the GUI can be turned off using a response file.
- Client: Downloaded via Java Webstart.

The Problem

- There’s a bunch of “official” servers for the build process (Jenkins).
- There’s a bunch of “development” virtual machine for the developers, roughly one for developer.
- We want to install our application from “official” servers to development servers.
- Launch integration tests after installation.

But...

The Environment



The (challenging) Environment

- No sudo access to Vogon's servers.
- Limited access to Jenkins: for some users only.
- Sudo access to "devservers" for every developer.
- NFS builds spool directory accessible by both groups: vogons and devservers.
- NFS dev tools directory accessible by both groups.

Doubts and concerns

- It is not possible to install applications on a Vogon's server (no Puppet master).
- Use Fabric?
- Use Chef or Saltstack?

Wish you were here

Ian Bicking

Shared publicly - Jul 6, 2013

I don't really know anything about Ansible, but from the most shallow of readings it seems interesting, and more importantly simple. Puppet and Chef are impossibly obtuse.

Shell scripts are also still pretty awesome, when paired with semi-disposable servers.

Ansible Simply Kicks Ass |

devo.ps

+14 4

Ansible 101

- No Agent, like Fabric, uses only ssh.
- Centralized configuration file.
- Pluggable, modular.
- Plenty of plugins (for less challenging environments).
- Supports groups and roles.
- YAML.

Ansible 102

- Comes with two command line tools for:
- Ad-hoc commands.
- Orchestration = Playbooks.

```
ansible -i ~/etc/hosts devservers -m ping
ansible -i ~/etc/hosts devservers -u root -a “cat
/etc/shadow”

ansible-playbook -i ~/etc/hosts devservers
install.yml
```

Playbook 101

- Is written in YAML.
- A playbook is a list of plays.
- A play is a list of tasks, played by a user on a group.

Tasks 101

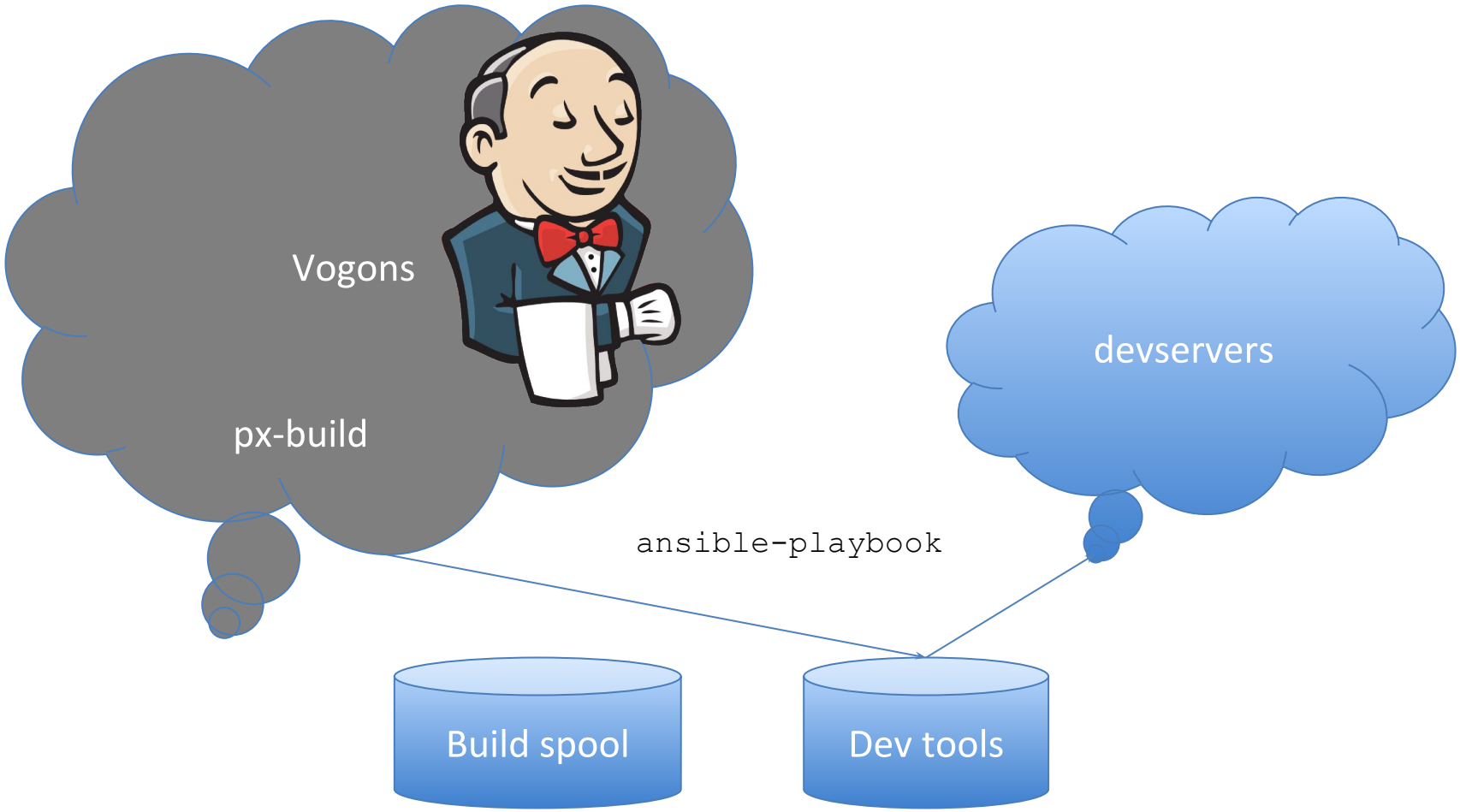
- A task can be defined in a playbook or in a task file written in YAML.
- A task file contains one ore more things to do.
- A task uses the Ansible’s plugin functionalities.
- A task uses Ansible’s variables.
- A task uses user defined variables.

Requirements

- Master: Python w/ Ansible on the host where ansible is running: a Vogon's server or another devservers.

- Target: a devserver w/ at least Python 2.6 or lesser with simplejson installed.

The Solution



The Solution

- “Hi, can you please generate a SSH key for px-build named ci_rsa and give us the public key?”

- “Sure!”

The Solution: hosts file

```
[devservers]
arthur
trillian

[devservers:vars]
ansible_python_interpreter=/sw/packages/python/2.7.1
/bin/python
launch_dir=/data/cdimage/Disk1
nfs_dir=/builds/cdimages/LATEST
target_build_dir=/data/cdimage
target_inst_dir=/cisco/PrimeOpticalServer
```

The Solution: install.yml

```
---
- hosts: devservers
  tasks:
  - include: tasks/checkinstalled.yml
  - include: tasks/remove-tar.yml
  - include: tasks/expand-tar.yml
  - include: tasks/response-file.yml

- hosts: devservers
  user: root
  tasks:
  - include: tasks/install.yml

- include: postinstall.yml
```

The solution: deploy.yml

```
---
- hosts: devservers
  user: root
  tasks:
  - include: tasks/check-tar-exists.yml
  - include: tasks/uninstall.yml

- include: install.yml
```

The Solution: checkinstalled.yml

An example of “if”.

```
---
# check if server already installed tasks
- name: check if server is installed
  command: ls {{ target_inst_dir }}
  register: result
  ignore_errors: True
- fail: msg="Server is already installed"
  when: result|success
```

The Solution: response-file.yml

Example of template module.

```
---
# prepare response file tasks
- name: remove response file
  command: rm -f /tmp/ctm.properties
- name: transfer installation response file
  template: src=./templates/ctm.properties.j2
dest=/tmp/ctm.properties
```

The Solution: jinja2 template

```
…
DB_IP_ADDRESS={{ ansible_eth0["ipv4"]["address"] }}
…
SERVER_HOSTNAME={{ ansible_fqdn }}
SERVER_IP_ADDRESS={{ ansible_eth0["ipv4"]["address"] }}
SUDO_GROUP=root
USER_INSTALL_DIR={{ target_inst_dir }}
```

The Solution: bash glue

ci.sh is a glue shell script invoked by Jenkins running on a Vogon's server as px-build user.

```
 #!/bin/sh

. /opt/dev-tools/python/Linux/2.7.4/python/bin/activate

(cd /opt/dev-tools/lazyjack/release/ansible &&
ansible-playbook --private-key=~/.ssh/ci_rsa -i hosts
deploy.yml)
```

Gotchas!

- Thank to Ansible, any host in the “devservers” group can install any other host of the same group.
- I’ve integrated Ansible with Jenkins to bypass the GUI installer.
- Now a Vogon’s server, which is also a Jenkins slave, can install any number of “deveservers”
- We can control which host to install using hosts file w/o accessing Jenkins.

Takeaways

- Ansible is very easy and well documented!
- Central configuration file is a win.
- Parallel by default (5 processes).
- Playbooks (orchestration) and configuration are separated.
- It’s very easy to reuse “tasks” and make them modular.
- Python plugins included (but I used a few).

Q&A

So Long, and Thanks for All the Fish.

(Douglas Adams)