Ansible workshop

The easiest way to:

orchestrate, deploy and manage

http://dag.wiee.rs/attic/ansible-workshop/

NLUUG Spring Conference, Utrecht, NL

Jeroen Hoekx, jeroen@hoekx.be
Dag Wieërs, dag@wiee.rs

Booming project

- **Started in February 2012**
  - Well-defined unique selling proposition
  - Small auditable codebase (back in the day ;-))
- **Very high activity**
  - 700+ unique contributors in first 2 years
  - Has slowed down when maturing
- **Divers use-cases / userbase**
  - large dotcoms, hosting providers, universities, banks, government agencies, consultants, startups, Open Source projects

Compelling features

- **Uses SSH, no agent**
  - Self-bootstrapping, no installation
  - No extraneous PKI, uses existing authentication/authorization
- **Swiss army knife**
  - *parallel execution, provisioning, application deployment, configuration management, orchestration, use-as-a-library, reporting tool, ...*
- **Complex orchestration made easy**
  - Simple management language (YAML-based !)
  - “Infrastructure as data” (not as code !)
  - Multi-tier management, multi-user
- **Core written in python**
  - Modules can be in any language, interfaces using JSON
- **Get started in less than 10 minutes**

Buzzword compliant

- **Privilege escalation**
  - sudo, su, powerbroker, …
- **Idempotency**
  - Not enforced, but advised
- **Orchestration**
  - Control “when”, “what” and “where”
- **Declarative**
  - Language limits complexity
- **Multi-user**
  - Power to the people !
- **Module development**
  - Any language supported by target *(python, powershell)*
- **Transports**
  - local, ssh, chroot, jail, lxc, winrm, zone, fireball, accelerate, funcd
- **Integration**
  - Design allows integration at various levels

Easy to get running

- **Requirements:**
  - python 2.6, paramiko/openssh, PyYAML, jinja2
- **Run or install from checkout**
  - `git clone git://github.com/ansible/ansible.git`
  - `cd ./ansible`
    1. `source ./hacking/env-setup`
    2. `make install`
- **Install distribution package or make your own**
    3. `make deb / make rpm`

Setting up demo environment

KVM and Libvirt

- Copy **vm-noname.img** to local disk-store (/var/lib/libvirt/images)
- Create new VM “vm-master”
  - Use “import existing disk image” but “Browse local” to vm-master.img
  - As a Linux guest using Red Hat EL6
  - Use 1 CPU and 512MB RAM
  - Use the “Virtual network 'default': NAT”
- Clone this VM as “vm-web
- Clone this VM again as “vm-db”
- Start all VMs

Virtual Box

- Copy **vm-noname.vmdk** to local disk
- Create a Host-Only network vboxnet0 and use it below
- Create new VM “vm-master”
  - As a Linux guest using Red Hat (32bit)
  - Use 512MB RAM
  - Use “an existing virtual hard drive file” (vmdk)
  - Modify the VM to use the created Host-Only network vboxnet0
- Clone this VM as “vm-web”
- Clone this VM again as “vm-db”
- Start all VMs

Everybody ready ?

- **During this session:**
  - Documentation available from:

    docs.ansible.com
  - Let us know if you need help

- **To proceed, log on to vm-master using SSH**
  - Username: root / Password: root
  - Go inside *~/workshop/*
  - Edit the *hosts* file
  - Use the IP addresses from the other VMs

Terminology

- **Inventory** – *flat file(s), yaml or custom scripts*
  - **Collection of groups, hosts, variables**
- **Modules** – *scripted in any language, using json*
  - **Offers specific functionality used in tasks**
- **Plugins** – *python scripts*
  - action, callback, connection, filter, lookup, ...
- **Playbooks** – *yaml description*
  - **Collection of plays**
    - **Collection of tasks**

Plethora of modules...

Action	assemble, command, copy, fetch, get_url, ping, raw, script, shell, slurp, template, uri
Management	authorized_keys, cron, file, group, ini_file, lineinfile, lvol, mount, seboolean, selinux, service, supervisorctl, sysctl, user, virt, zfs
Deployment	cloudformation, django_manage, easy_install, fireball, gem, git, hg, mongodb_user, mysql_db, mysql_user, nagios, pip, postgresql_db, postgresql_user, rabbitmq_parameter, rabbitmq_plugin, rabbitmq_user, rabbitmq_vhost, subversion
OS specific	apt, apt_key, apt_repository, macports, opkg, pacman, pkgin, svr4pkg, yum
Workflow	add_host, async_status, debug, fail, group_by, mail, pause, wait_for
Inventory	ec2_facts, facter, hpilo_facts, network_facts, ohai, setup, virt_facts, vsphere_facts
Provisioning	ec2, ec2_vol, hpilo_boot, virt_boot, virt_create, vsphere_boot

Ansible troubleshooting

- **Actions:** *Increase verbosity*

  -v Display JSON module output

  -vv Display (real) targets / communication

  -vvv Display low-level SSH execution

  -vvvv Display SSH verbose communication

- **Modules:** *Test individual modules remotely*

```
export ANSIBLE_KEEP_REMOTE_FILES=1
```

- **Delays:** *Use “pstree” on remote ends*

```
watch -n1 'for pid in $(pgrep sshd); do pstree -al $pid; done'
```

- **Freezes:** *Disable pty's to avoid input prompts (paramiko)*
- **Logic:** *Add debug actions to print data structures*
- **Templates:** *Use --check and --diff during development*

Ansible tips and tricks

- **The “action: module” dilemma**
  - Don't be fooled, YAML tasks ***are*** dictionaries (!)
- **Playbooks are “documented” declarations**
  - Always name your actions
  - Don't describe, but give meaning
- **Keep playbooks simple and honest**
  - Use dynamic inventories to state context
  - Templates can help to reduce playbook spaghetti
  - Push complex logic into custom modules (locality)
  - Sometimes custom lookup_plugins and with_* can help
- **Idem-potency is key ! Modules can help, but...**
  - Use “creates=” and “removes=” where possible
  - Use “changed_when:” and “failed_when:” to influence outcome

Join in on the fun !

- Learn more at:

  docs.ansible.com

- Talk to us on IRC at:

  #ansible on Freenode.net

- Discuss on the Ansible mailing list at:

  groups.google.com/group/ansible-project

- Find us on GitHub at:

  github.com/ansible/ansible

Thank you for listening !

This workshop is available from:
github.com/ansible-provisioning