

IT Automation with Ansible



Dr. Bernhard Hopfenmüller

19.10.2017

How complicated is complicated enough?



<http://www.hldataprotection.com/2010/09/articles/consumer-privacy/if-the-online-notice-is-too-complex-does-that-open-the-door-to-tort-claims/>



<https://depositphotos.com/16900495/stock-photo-good-luck.html>

...if it is too complicated?

Ansible ...

- ▶ ... is „radically simple“
- ▶ ... requires: 1 Unixnode, SSH, Python
- ▶ ... needs no agent
- ▶ ... describes the desired state
- ▶ ... is written in YAML
- ▶ ... satisfies idempotency



- ▶ User: Write Statements
- ▶ Ansible: Create program
- ▶ Ansible: Copy program to Run-Node
- ▶ Ansible: Connect to Run-Node (local, SSH, PowerShell)
- ▶ Ansible: Run Program
- ▶ Ansible: Communication via JSON
- ▶ Ansible: Delete local Program
- ▶ Ansible: Give Feedback to User



- ▶ **Inventory** - infrastructure as code
- ▶ **Adhoc mode** - one task + terminal
- ▶ **Playbook** - collection of tasks/roles
- ▶ **Role** - one collection of tasks
- ▶ **Task** - one Ansible command:
'apt-get install nano'
- ▶ **Jinja 2 templating** - e.g. for variables
- ▶ **Modules** - core-components:
yum module, ping module,...
communication via JSON

```
[dbserver]
db1.server.com
db2.server.com

[webserver]
web1.server.com ansible_host=10.11.12.13
web2.server.com ansible_host=10.11.12.14

[appserver]
app1.server.com default_port=9418
app2.server.com default_port=42

[linuxserver:children]
dbserver
webserver
appserver
```

- ▶ defining the infrastructure
- ▶ ini or yaml syntax
- ▶ combine servers into groups
- ▶ define specific vars

- ▶ run only one task quickly
- ▶ either one-timer or quicker than playbook

```
$ ansible webserver -m patch \
-a \
"src=/src/critical.patch \
dest=/app/bin/base"

$ ansible webserver -a "sbin/reboot"

$ ansible webserver -m ping
```

```
---  
- hosts: webservers # where shall I run?  
  become: true      # run as root  
  roles:  
    - bake_webserver # role 1  
    - install_gitlab # role 2
```

- ▶ combination of roles
- ▶ written in YAML (or JSON)-Syntax
- ▶ ideally in combination with VCS

Setup a gitlab server

```
- name: Install GitLab dependencies.
  yum:
    name: {{ item }}
    state: installed
  with_items:
    - openssh-server
    - postfix
    - curl
    - openssl

- name: Install GitLab
  yum:
    name: {{ gitlab_edition }}
    state: present

- name: Copy GitLab configuration file.
  template:
    src: gitlab.rb.j2
    dest: /etc/gitlab/gitlab.rb
    owner: root
    group: root
    mode: 0600
  notify: restart gitlab
```

- ▶ combination of tasks
- ▶ featuring idempotency
- ▶ ideally in combination with VCS

```
- name: Install GitLab dependencies.
  yum:
    name: {{ item }}
    state: installed
  with_items:
    - openssh-server
    - postfix
    - curl
    - openssl

- name: Install GitLab
  yum:
    name: {{ gitlab_edition }}
    state: present

- name: Copy GitLab configuration file.
  template:
    src: gitlab.rb.j2
    dest: /etc/gitlab/gitlab.rb
    owner: root
    group: root
    mode: 0600
  notify: restart gitlab
```

ATIX and Ansible - a truly great combination



Engineering for and with Ansible



ANSIBLE

+



FOREMAN


- ▶ Active Member of the OpenSource Community
- ▶ Ansible Modules for Foreman (amongst main contributors)
- ▶ More modules under development
- ▶ Ansible as daily QA for orcharhino

Configuration Management for lazy smart DevOps

- ▶ convenient and easy setup
- ▶ setup 100 % reproducible
- ▶ any combination is possible
- ▶ easily extendable



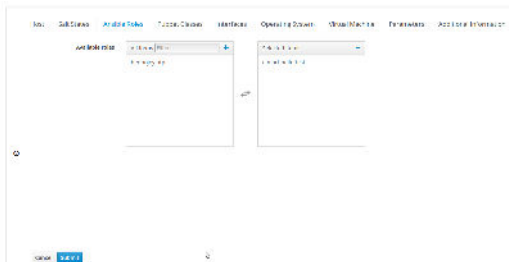
Webinstaller



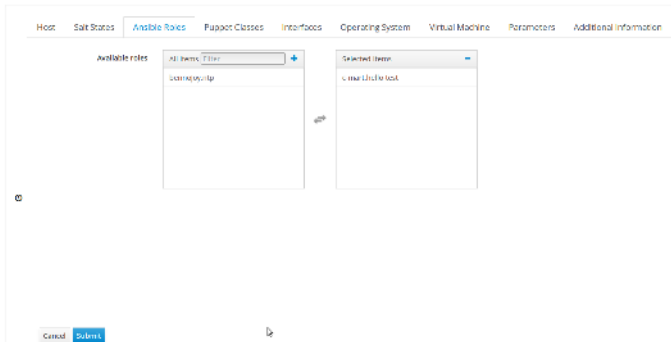
The screenshot shows the 'orcharhino web installer' interface. On the left is a vertical sidebar with six numbered steps: 1. Basic configuration (Basic configuration), 2. Networking capabilities (Network & Storage), 3. HTTP Proxy (Configure HTTP Proxy), 4. Operating systems (Create Operating System), 5. Configuration management (Manage Files/Profiles), and 6. Plugin selection (Configure additional services). Step 4 is currently selected and highlighted in blue. The main content area is titled 'Operating systems' and displays a list of operating system options with checkboxes: CentOS 6, CentOS 7, Debian 7, Debian 8, Ubuntu Server 12.04, Ubuntu Server 14.04, and Ubuntu 14.04. At the bottom of the main area are three buttons: 'Previous', 'Next', and 'Cancel'.

- ▶ Installer for orcharhino
- ▶ Configure according to your needs
- ▶ Installer writes config
- ▶ Runs playbook
- ▶ Idempotency ↔ Customize and rerun

Use Ansible within orcharhino



- ▶ Run playbooks from orcharhino
- ▶ in combination with Puppet or standalone
- ▶ under active development



The screenshot shows the ATIX web interface for configuring Ansible roles. At the top, there is a navigation bar with tabs: Host, Salt States, **Ansible Roles**, Puppet Classes, Interfaces, Operating System, Virtual Machine, Parameters, and Additional Information. Below the navigation bar, the 'Available roles' section contains a search dropdown menu with 'All items: 1 item' and a plus sign. Below the search bar, the text 'salt-ansible' is visible. To the right of the available roles is a 'Selected items' section with a minus sign, containing the text 'salt-ansible test'. An arrow points from the available roles list to the selected items list. At the bottom left of the interface, there are two buttons: 'Cancel' and 'Submit'.

Task Running Steps Errors Locks Raw

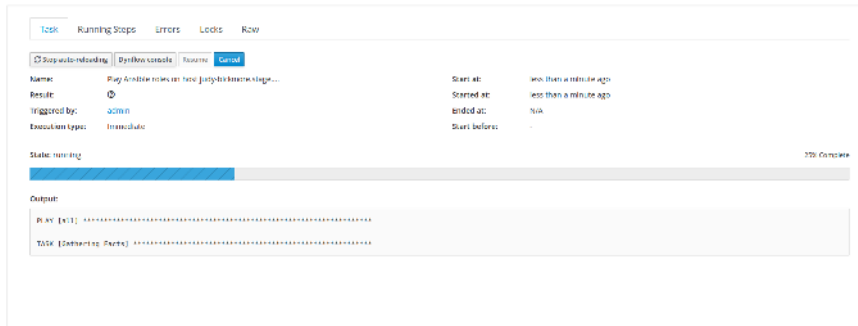
Stop auto-reloading DRYflow console Freeze Cancel

Name:	Play Ansible roles on host judy beckmeier.stage...	Start at:	less than a minute ago
Result:	🟢	Started at:	less than a minute ago
Triggered by:	admin	Ended at:	now
Execution type:	Immediate	Start before:	-

Status running 0% Complete

Output:


```
PLAY [a:1] *****
TASK [athering facts] *****
```



The screenshot shows the 'Task' tab of an Ansible play execution. The task is 'Play Ansible roles on host judy@rhino.stage...'. The status is 'running', indicated by a blue progress bar. The 'Output' section shows the start of the play and task execution.

Task: Running Steps Errors Logs Raw

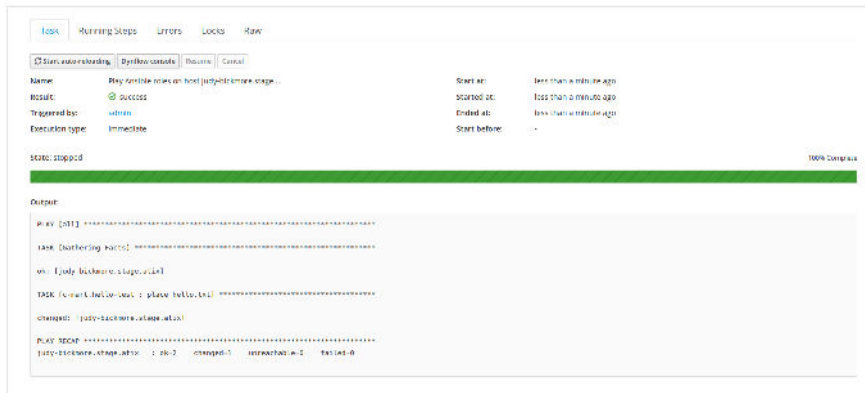
Step into role: Python scripts Runners General

Name: Play Ansible roles on host judy@rhino.stage... Start at: less than a minute ago
Result:  Started at: less than a minute ago
Triggered by: admin Ended at: N/A
Execution type: Immediate Start before: -

State: running 37% Complete

Output

```
PLAY [all] *****  
TASK [Gathering Facts] *****
```



The screenshot displays the ATIX web interface for an Ansible task. At the top, there are tabs for 'Task', 'Running Steps', 'Errors', 'Locks', and 'Raw'. Below these are buttons for 'Start auto-reloading', 'Dyfflow console', 'Home', and 'Cancel'. The main content area shows task details:

- Name:** Play ansible roles on host juty-backere.stage...
- Result:** Success (indicated by a green checkmark)
- Triggered by:** admin
- Execution type:** Immediate
- Start at:** less than a minute ago
- Started at:** less than a minute ago
- Ended at:** less than a minute ago
- Start before:** -

Below the details, it shows 'state: stopped' and a green progress bar indicating '100% Complete'. The 'Output' section contains the following text:

```
PLAY [all] *****
TASK [gathering facts] *****
ok: [juty-backere.stage.atix]
TASK [setup:hello-test : play hello, test] *****
changed: [juty-backere.stage.atix]
PLAY RECAP *****
juty-backere.stage.atix : ok=2  changed=1  unreachable=0  failed=0
```


Example projects



- ▶ (one time) Deployment of infrastructures
- ▶ interface to virtualization
- ▶ CI - CD integration
- ▶ VCS integration
- ▶ Springboot framework
- ▶ Compliance Management
- ▶ ...

- ▶ From end of 2017, Begin of 2018
- ▶ 3 days
- ▶ Hands-on based
- ▶ topics (can) include
 - ▶ Ansible basics
 - ▶ From Paper to Playbook
 - ▶ Jinja Templating
 - ▶ Ansible + VCS
 - ▶ (Tower)



Ansible - Simple and versatile IT Automation



- ▶ Engineering
- ▶ Consulting
- ▶ Training
- ▶ Support