SCALEOUT

Configuration Management Tools

Max Andersson

Ansible

What Ansible aims to be

- Clear/ simple api’s
- Fast to learn & setup
- Complete
- Efficient
- Secure
    - Uses SSH by default

Ansible

Finding more information on Ansible

- Documentation
- Glossary
- Mailing List
- Github
- Blog
- Example Playbooks

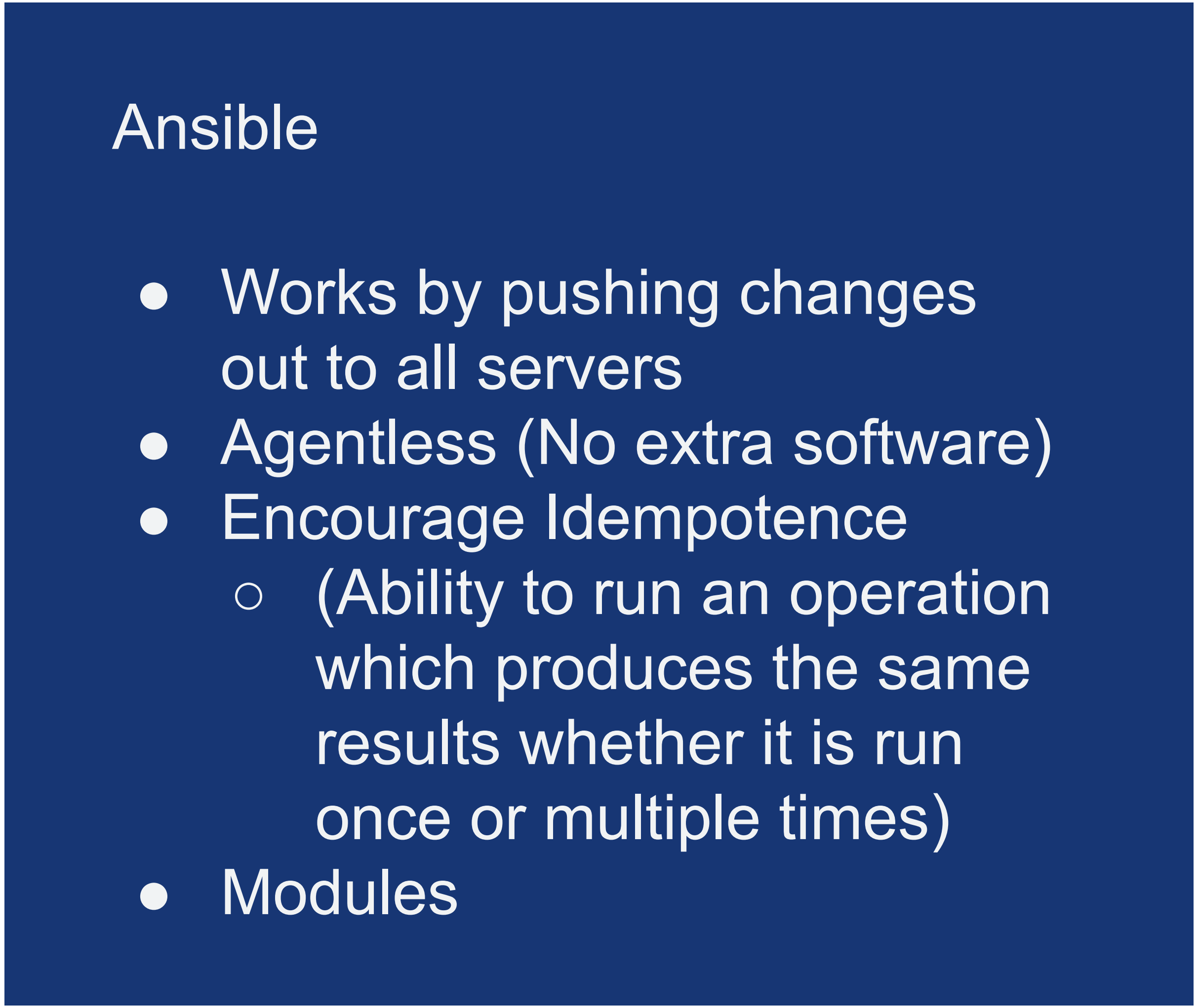
Ansible

Conducting an Orchestra

The Orchestra

- Patches and updates
- Resource Usage
- Checking Logs
- Manage users and groups
- Dns settings, hostfiles, etc..
- Deploy and run apps
- Manage cron-jobs

Ansible



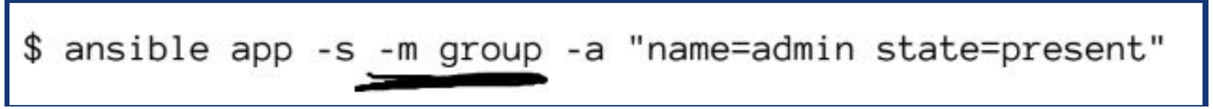
Ansible

- Works by pushing changes out to all servers
- Agentless (No extra software)
- Encourage Idempotence
  - (Ability to run an operation which produces the same results whether it is run once or multiple times)
- Modules

Ansible

Modules

- Predefined modules such as
  - users
  - group
  - package
  - start
  - copy
  - fetch
  - file

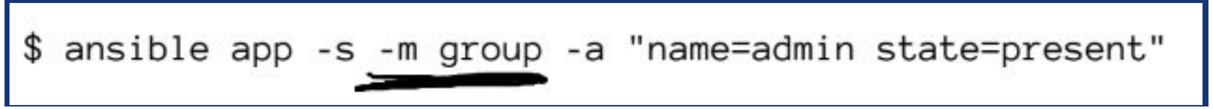


```
$ ansible app -s -m group -a "name=admin state=present"
```

Ansible

Modules

- Community modules such as
  - Cloud modules
  - File modules
  - Git modules
  - Crypto modules
  - Network modules
  - etc..



```
$ ansible app -s -m group -a "name=admin state=present"
```

Ansible

Common options

 --sudo, -s (or specify become: true in playbook)
 --ask-sudo-pass, -K

Ansible

Inventory file vs. Dynamic inventory

Ansible

Ansible

Inventory

- Hosts and groups
- Host and group variables
- Group of groups
- Default groups
  - all
  - Ungrouped
- Priority (if not otherwise specified)
  - all group
  - parent group
  - child group
  - host
- Behavioral paramaters

Ansible

Inventory file

```
##/etc/ansible/hosts

192.0.2.50
aserver.example.org
bserver.example.org
```

Inventory file

Ansible

ini

```
##/etc/ansible/hosts

mail.example.com

[webservers]
foo.example.com
bar.example.com

[dbservers]
One.example.com
two.example.com
three.example.com
```

yaml

```
all:
  hosts:
    mail.example.com:
  children:
    webservers:
      hosts:
        foo.example.com:
        bar.example.com:
    dbservers:
      hosts:
        one.example.com:
        two.example.com:
          three.example.com:
```

Inventory file

Ansible

```
##/etc/ansible/hosts

[atlanta]
host1
host2

[atlanta:vars]
ntp_server=ntp.atlanta.exampl
e.com
  proxy=proxy.atlanta.example.
  com
```

```
##/etc/ansible/hosts

  atlanta:
    hosts:
      host1:
      host2:
    vars:
      ntp_server:
  ntp.atlanta.example.com
      proxy:
  proxy.atlanta.example.com
```

Dynamic Inventory

Ansible

- When Hosts are not static.
- You need multiple sources
- Need to integrate with LDAP or likewise.
- Want to integrate with a service provider such as Openstack, AWS or GCP though plugins

Ad-Hoc Commands

- Can execute commands on multiple hosts in parallel or in serially.
- Can be used to check status from multiple machines
- Uses SSH. Assumes keys are in the right place and uses passwordless auth, otherwise you must provide --ask-pass flag

Playbooks

Ansible

- Playbooks refers to strategies in american football, and specific implementations are referenced as plays.
- In ansible playbooks define configuration, deployments etc..
- Can use roles to group plays that need to be repeated

Playbooks

Ansible

- Static and Dynamic imports/includes
  - Static
    - Pre-processes static import when parsing
    - Cascades to child tasks
  - Dynamic
    - Dynamic includes are processed during runtime
    - Does not cascade to child tasks

Playbooks

- Variables
- Templating
- Conditionals
- Loops
- Blocks

Best Practices

- Directory layout
- Staging vs Production
- Rolling updates
- Use Roles for groups to keep it DRY
- Always Name Tasks
- USE VERSION CONTROL!

Ansible

```
production                # inventory file for production servers
staging                   # inventory file for staging environment

group_vars/
   group1.yml             # here we assign variables to particular groups
   group2.yml
host_vars/
   hostname1.yml          # here we assign variables to particular systems
   hostname2.yml

library/                  # if any custom modules, put them here (optional)
module_utils/             # if any custom module_utils to support modules, put
them here (optional)
filter_plugins/           # if any custom filter plugins, put them here
(optional)

site.yml                  # master playbook
webservers.yml            # playbook for webserver tier
dbservers.yml             # playbook for dbserver tier
```

Ansible

```
roles/
    common/               # this hierarchy represents a "role"
        tasks/            #
            main.yml      #  <-- tasks file can include smaller files if warranted
        handlers/         #
            main.yml      #  <-- handlers file
        templates/        #  <-- files for use with the template resource
            ntp.conf.j2   #  <------- templates end in .j2
        files/            #
            bar.txt       #  <-- files for use with the copy resource
            foo.sh        #  <-- script files for use with the script resource
        vars/             #
            main.yml      #  <-- variables associated with this role
        defaults/         #
            main.yml      #  <-- default lower priority variables for this role
        meta/             #
            main.yml      #  <-- role dependencies
        library/          # roles can also include custom modules
        module_utils/     # roles can also include custom module_utils
        lookup_plugins/   # or other types of plugins, like lookup in this case

    webtier/              # same kind of structure as "common" was above, done for
the webtier role
    monitoring/           # ""
      fooapp/               # ""
```

1. Infrastructure as code — Ansible, Terraform, Puppet, Chef
2. CI/CD — Jenkins, TeamCity, Shippable, Bamboo, Azure DevOps
3. Test automation — Selenium, Cucumber, Apache JMeter
4. Containerization — Docker, Rocket, Unik
5. Orchestration — Kubernetes, Swarm, Mesos
6. Software deployment — Elastic Beanstalk, Octopus, Vamp
7. Measurement — NewRelic, Kibana, Datadog, DynaTrace
8. ChatOps — Hubot, Lita, Cog

1.