ANSIBLEWORKS

Radically simple IT orchestration

Quick-Start

Your Instructor:

Tim Gerla

tim@ansibleworks.com

Twitter: @Tybstar

Questions during the presentation?

Join the #ansible IRC channel on FreeNode:

http://webchat.freenode.net/

What is Ansible?

- Orchestration
- Software Deployment
- Configuration Management

ANSIBLEWORKS

How is it different?

- No custom PKI–SSH-based
- Agentless architecture
- Configuration as data, not code
- Batteries-included
- Full configuration management, orchestration, and deployment

Quick-Start Outline

- Ansible Basics
- Host Inventory
- Playbooks
- Modules
- Variables
- Full Example Walkthrough

An Example

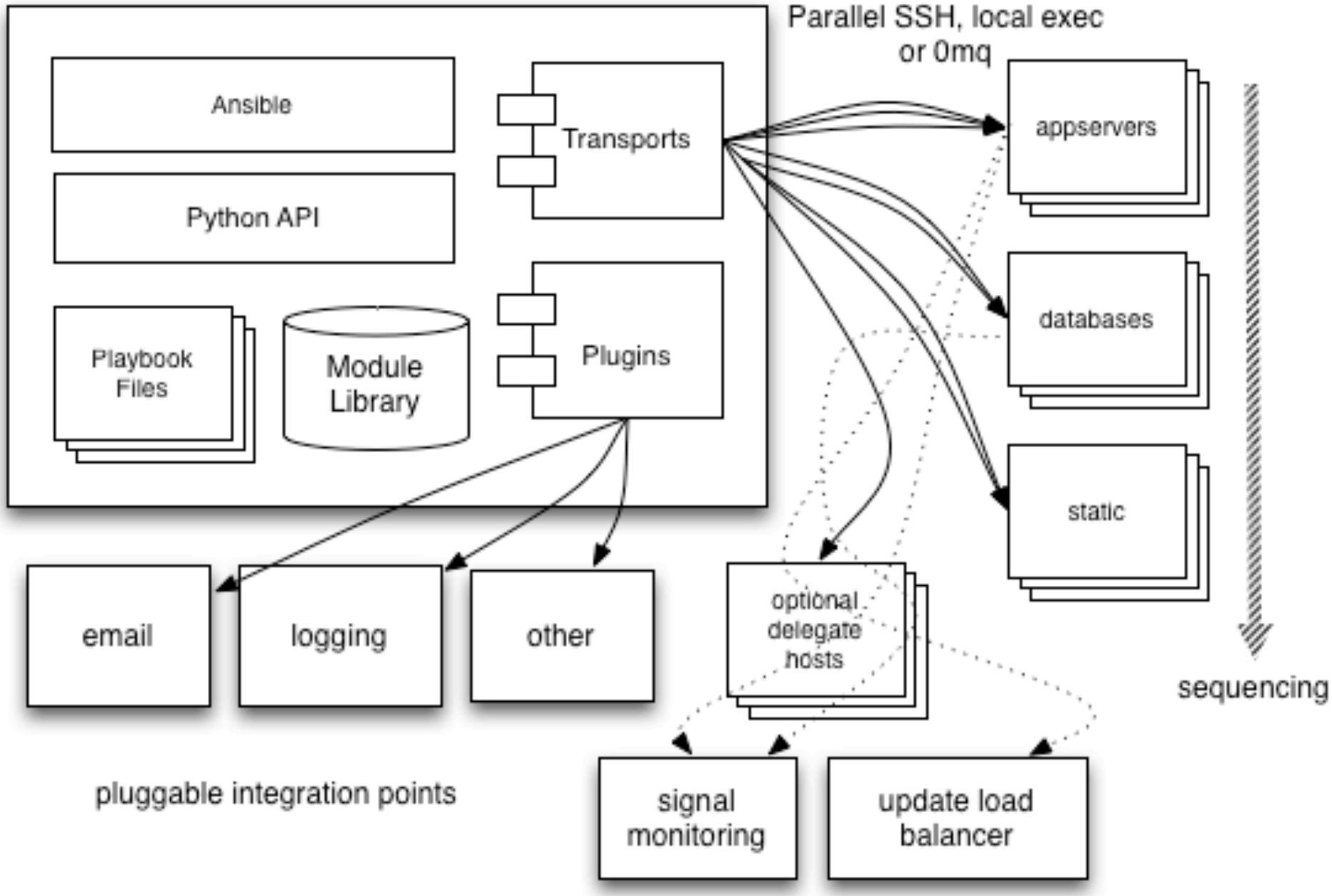
```
---
- name: install and start apache
  hosts: all
  user: root

  tasks:

  - name: install httpd
    yum: name=httpd state=present

  - name: start httpd
    service: name=httpd state=running
```

Ansible Architecture



ANSIBLEWORKS

Install and Configure

- Your choice:
  - From Git (recommended for the examples in this presentation)
  - From Packages
  - From PIP

http://ansible.cc/docs/gettingstarted.html

Demo

- Ad-Hoc command example
- Basic playbook example

Host Inventory: Basics

```
[web]
webserver-1.example.com
webserver-2.example.com
```

```
[db]
dbserver-1.example.com
```

Host Inventory: Ranges

```
[web]
webserver-[01:25].example.com

[db]
dbserver-[a:f].example.com
```

Host Inventory: More

- Non-standard SSH ports:

```
webserver.example.com:2222
```

- SSH tunnel:

```
myhost ansible_ssh_port=5555
 ansible_ssh_host=192.168.0.1
```

Inventory: child groups

```
[east]
host1
host2

[west]
host3
host4

[us:children]
east
west
```

Connection Types

- paramiko (Python SSH module)
- ssh
- local
- chroot

Ansible Concepts

- Playbooks
- Plays
- Tasks and handlers
- Modules
- Variables

ANSIBLEWORKS

Playbooks

- Playbooks contain Plays
  - Plays contain Tasks
    - Tasks call Modules
- Everything is sequentially ordered–strict dependency ordering. Handlers can be triggered by tasks, and will run at the end, once.

Tasks

- A *task* calls a *module* and may have parameters. Ansible has a lot of modules included, and you can write your own.

```
tasks:
- name: ensure apache is at the latest version
  yum: name=httpd state=latest

- name: write the apache config file
  template: src=templates/httpd.j2 dest=/etc/httpd.conf

- name: ensure apache is running
  service: name=httpd state=started
```

Modules

- Ansible is “batteries included”:

```
add_host
apt
apt_key
apt_repository
assemble
async_status
async_wrapper
authorized_key
bzr
cloudformation
command
copy
cron
debug
django_manage
easy_install
ec2
ec2_facts
ec2_vol
facter
fail
fetch
file
fireball
gem
get_url
git
group
group_by
hg
homebrew
ini_file
lineinfile
lvg
lvol
macports
mail
mongodb_user
mount
mysql_db
mysql_user
nagios
netscaler
ohai
openbsd_pkg
opkg
pacman
pause
ping
pip
pkgin
postgresql_db
postgresql_user
rabbitmq_parame
ter
rabbitmq_plugin
rabbitmq_user
rabbitmq_vhost
raw
rhn_channel
s3
script
seboolean
selinux
service
setup
shell
slurp
subversion
supervisorctl
svr4pkg
sysctl
template
uri
user
vagrant
virt
wait_for
yum
zfs
```

Modules, Continued

- Package management: **yum, apt**
- Remote execution: **command, shell**
- Service management: **service**
- File handling: **copy, template**
- SCM: **git, subversion**

command and *shell*

- Execute arbitrary commands on remote hosts.

```
- name: turn off selinux
  command: /sbin/setenforce 0

- name: ignore return code
  shell: /usr/bin/somecommand && /bin/true
```

- Long lines can wrap:

```
- name: Copy ansible inventory file to client
  copy: src=/etc/ansible/hosts
        dest=/etc/ansible/hosts
        owner=root group=root mode=0644
```

copy and *template*

- Copy a file from Ansible host to managed host:

```
- name: copy a file
  copy: src=files/ntp.conf dest=/etc/ntp/ntp.conf
        owner=root group=root mode=0644
```

- Evaluate a *Jinja2* template:

```
- name: Copy ansible inventory file to client
  template: src=templates/motd
            dest=/etc/motd
            owner=root group=root mode=0644
```

apt and *yum*

- Package management:

```
- name: install httpd
  yum: name=httpd state=present

- name: install httpd
  apt: name=httpd=2.0 state=present
```

- Install a set of packages in one transaction:

```
- name: install a set of packages
  yum: name={{ item }} state=present
  with_items:
    - httpd
    - php
    - git
    - mysql-client
```

A Playbook

```
---
- name: install and start apache
  hosts: all
  user: root

  tasks:

  - name: install httpd
    yum: name=httpd state=latest

  - name: start httpd
    service: name=httpd state=running
```

Playbook

Play

Tasks

ANSIBLEWORKS

```
---
- name: webserver configuration play
  hosts: webservers

  vars:
    http_port: 80
    max_clients: 200

  tasks:
  - name: ensure that apache is installed
    yum: name=httpd state=present

  - name: write the apache config file
    template: src=httpd.j2 dest=/etc/httpd.conf
    notify:
    - restart apache

  - name: ensure apache is running
    service: name=httpd state=started

  handlers:
    - name: restart apache
      service: name=httpd state=restarted
```

hosts to target in this play

```
---
- name: webserver configuration play
  hosts: webservers

  vars:
    http_port: 80
    max_clients: 200

  tasks:
  - name: ensure that apache is installed
    yum: name=httpd state=present

  - name: write the apache config file
    template: src=httpd.j2 dest=/etc/httpd.conf
    notify:
    - restart apache

  - name: ensure apache is running
    service: name=httpd state=started

  handlers:
    - name: restart apache
      service: name=httpd state=restarted
```

variables

```
---
- name: webserver configuration play
  hosts: webservers

  vars:
    http_port: 80
    max_clients: 200

  tasks:
  - name: ensure that apache is installed
    yum: name=httpd state=present

  - name: write the apache config file
    template: src=httpd.j2 dest=/etc/httpd.conf
    notify:
    - restart apache

  - name: ensure apache is running
    service: name=httpd state=started

  handlers:
    - name: restart apache
      service: name=httpd state=restarted
```

tasks in this play

```
---
- name: webserver configuration play
  hosts: webservers

  vars:
    http_port: 80
    max_clients: 200

  tasks:
  - name: ensure that apache is installed
    yum: name=httpd state=present

  - name: write the apache config file
    template: src=httpd.j2 dest=/etc/httpd.conf
    notify:
    - restart apache

  - name: ensure apache is running
    service: name=httpd state=started

  handlers:
    - name: restart apache
      service: name=httpd state=restarted
```

```
---
- name: webserver configuration play
  hosts: webservers

  vars:
    http_port: 80
    max_clients: 200

  tasks:
  - name: ensure that apache is installed
    yum: name=httpd state=present

  - name: write the apache config file
    template: src=httpd.j2 dest=/etc/httpd.conf
    notify:
    - restart apache

  - name: ensure apache is running
    service: name=httpd state=started

  handlers:
    - name: restart apache
      service: name=httpd state=restarted
```

the name of an individual task

```
---
- name: webserver configuration play
  hosts: webservers

  vars:
    http_port: 80
    max_clients: 200

  tasks:
  - name: ensure that apache is installed
    yum: name=httpd state=present

  - name: write the apache config file
    template: src=httpd.j2 dest=/etc/httpd.conf
    notify:
    - restart apache

  - name: ensure apache is running
    service: name=httpd state=started

  handlers:
    - name: restart apache
      service: name=httpd state=restarted
```

the task's action

```
---
- name: webserver configuration play
  hosts: webservers

  vars:
    http_port: 80
    max_clients: 200

  tasks:
  - name: ensure that apache is installed
    yum: name=httpd state=present

  - name: write the apache config file
    template: src=httpd.j2 dest=/etc/httpd.conf
    notify:
    - restart apache

  - name: ensure apache is running
    service: name=httpd state=started

  handlers:
    - name: restart apache
      service: name=httpd state=restarted
```

another task

```
---
- name: webserver configuration play
  hosts: webservers

  vars:
    http_port: 80
    max_clients: 200

  tasks:
  - name: ensure that apache is installed
    yum: name=httpd state=present

  - name: write the apache config file
    template: src=httpd.j2 dest=/etc/httpd.conf
    notify:
    - restart apache

  - name: ensure apache is running
    service: name=httpd state=started

  handlers:
    - name: restart apache
      service: name=httpd state=restarted
```

we call the handler here

the handler itself

Playbook Demo

Playbooks Continued

- Variables
- Roles

Variables

- There are several sources for variables:
  - Playbooks
  - Inventory (group vars, host vars)
  - Command line
  - Discovered variables (**facts**)

Variables

- You can use variables in action lines:

```
---
- hosts: webservers
  vars:
    vhost: myhost.com

  tasks:
  - name: create a virtual host file for {{ vhost }}
    template: src=vhost.j2 dest=/etc/httpd/conf.d/{{ vhost }}

  - name: do something against {{ inventory_hostname }}
    command: echo “I’m on {{ inventory_hostname }}”
```

Facts

- Discovered **variables** about systems
- Some examples:

```
    "ansible_os_family": "RedHat",
    "ansible_distribution": "CentOS",
    "ansible_hostname": "webserver1",
    "ansible_default_ipv4": {
        "address": "172.16.183.141",
        "alias": "eth0",
        ...
    }
```

```
ansible -m setup hostname
```

Using Variables

- In a playbook:

```
tasks:
- name: report this machine's IP
  command: echo "My IP is {{ ansible_default_ipv4.address }}"
```

- In a template:

```
This is a template file, evaluated and then sent to the target machine.

This machine's IP address is {{ ansible_default_ipv4.address }}
```

```
---
# Variables for the HAproxy configuration

# HAProxy supports "http" and "tcp".
mode: http

# Port on which HAProxy should listen
listenport: 8888

# A name for the proxy daemon, this will be the
# suffix in the logs.
daemonname: myapplb

# Balancing algorithm:
balance: roundrobin

# Ethernet interface for haproxy
iface: '{{ ansible_default_ipv4.interface }}'
```

Playbook Demo

Roles

- Project organizational tool
- Reusable components
- Defined filesystem structure

Roles

```
webserver/
├── files
|   ├── epel.repo.j2
|   └── RPM-GPG-KEY-EPEL-6
├── handlers
|   └── main.yml
├── tasks
|   └── main.yml
└── templates
    └── httpd.conf.j2
```

Playbook Demo

- Roles

Rolling Updates

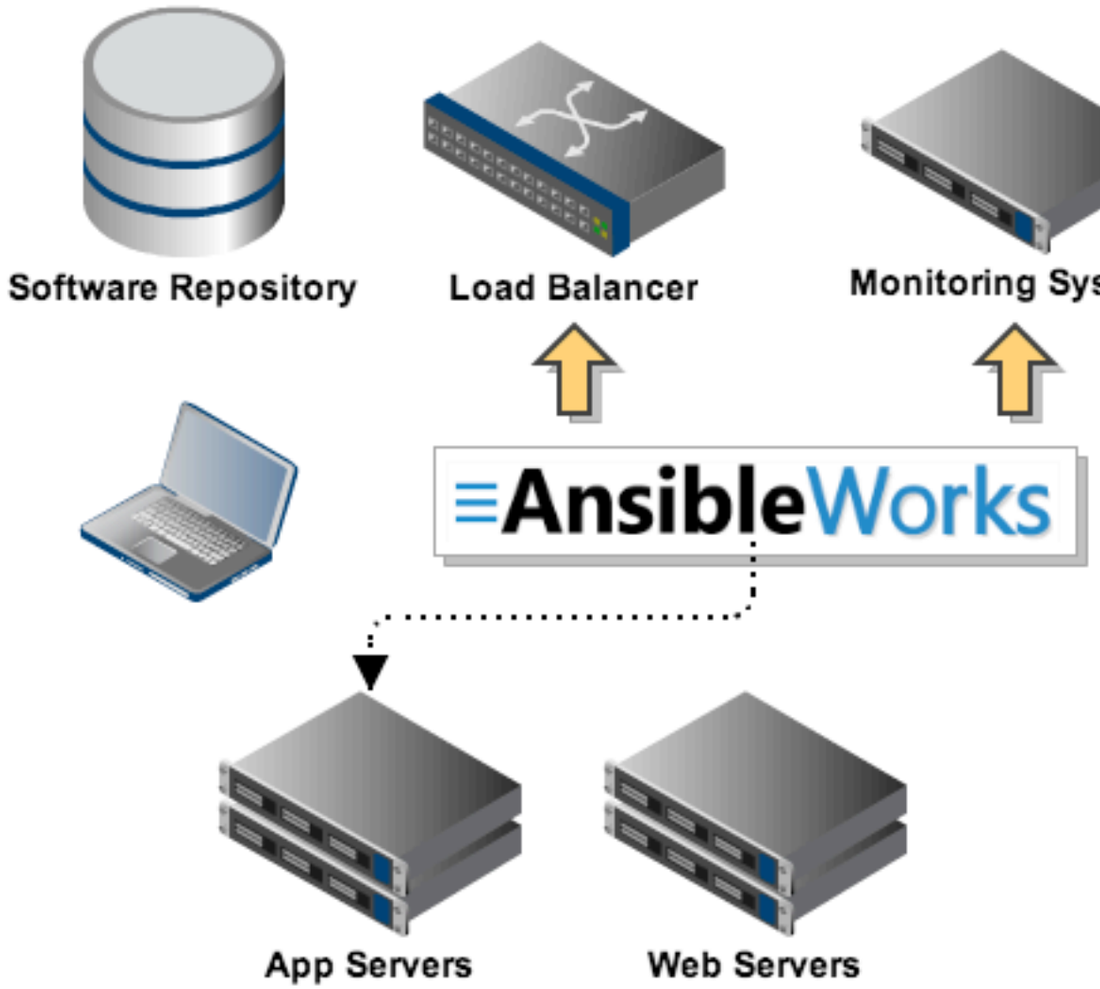
- *Serial* Keyword

Orchestration Example

The user executes an Ansible *playbook* which contains step-by-step instructions on how to perform the update.

Playbooks are simple, human-readable descriptions of IT workflows.

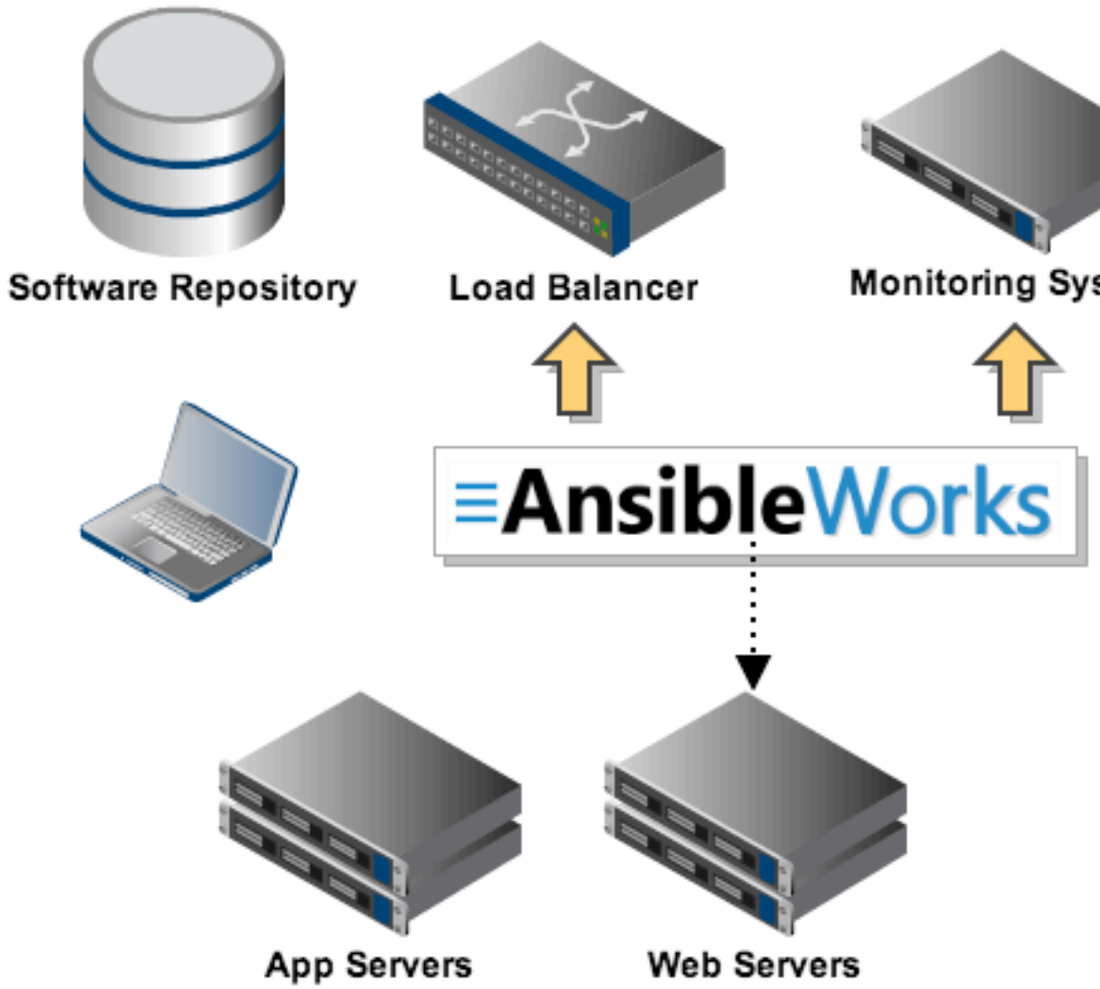
Orchestration Example



This step updates the app server tier. The servers are going to be updated ten at a time, in a rolling update.

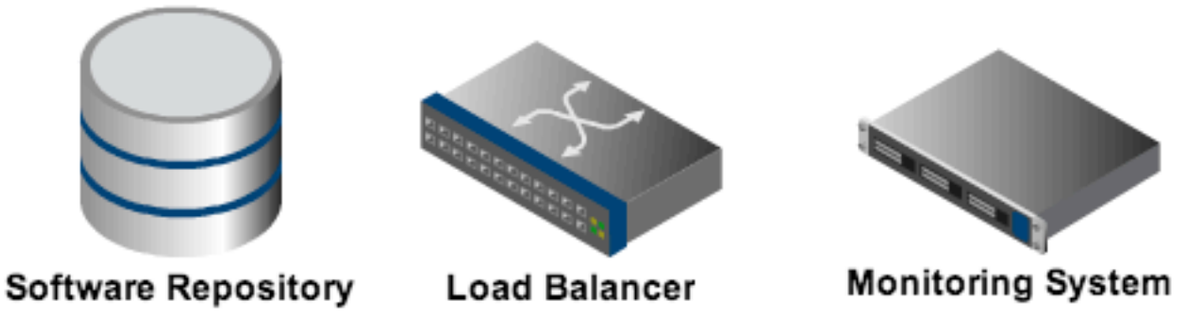
Ansible will talk to the load balancers to coordinate each batch. A maintenance window is also set on the monitoring system.

Orchestration Example



Now the same thing happens for the web server tier.

Orchestration Example



Finally, Ansible returns a summary of tasks performed.

Ansible can be configured to store data in a wide variety of data sources, send email reports, and more. If any errors occurred during any of the steps, the administrator will be notified.

AnsibleWorks

A Full Example

- Orchestration of a multi-tier web application

ANSIBLEWORKS

AnsibleFest!

- Inaugural Ansible users and developers conference
- Thursday, June 13, Boston, Mass.
- Save 20% off a ticket with code QUICKSTARTER

http://www.ansibleworks.com/fest/

AnsibleWorks

- Consulting/Training services available
- Beta of AnsibleWorks Suite coming soon!

ANSIBLEWORKS

Next Steps

- Documentation
- Example Playbooks
- IRC
- Mailing List and Newsletter

http://www.ansibleworks.com/

http://ansible.cc/

ANSIBLEWORKS

Q&A