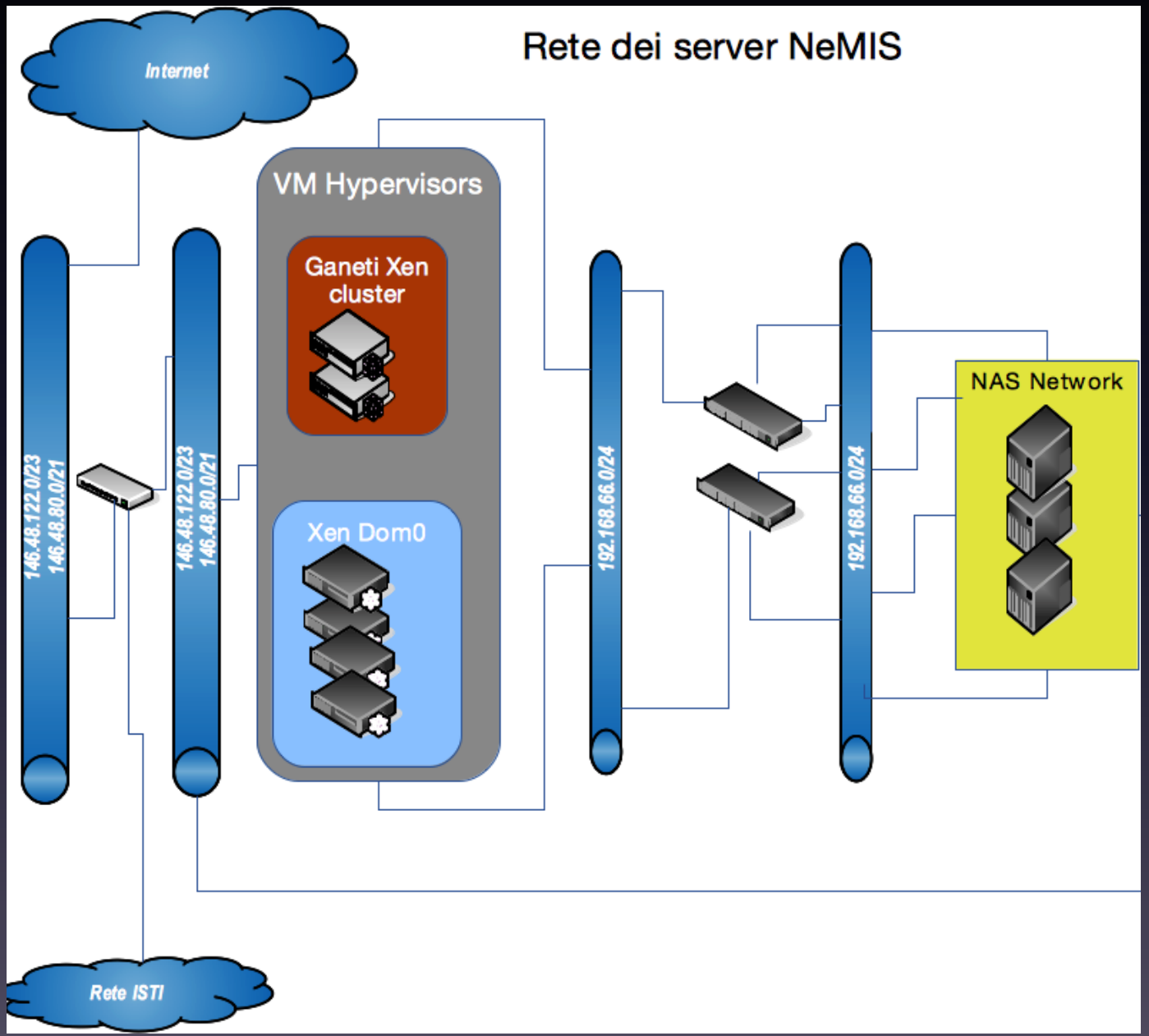
An overview of the NeMIS IT facilities

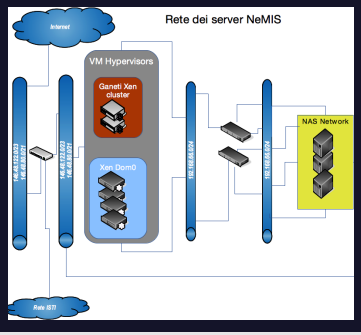
Andrea Dell'Amico

Room A-27, ISTI-CNR, July 2015

Table of Contents

NeMIS Infrastructure





The ISTI and NeMIS servers networks are not separated. The data center network and the offices network are not separated. This implies that each problem on the ISTI network (broadcast traffic out of control, vulnerable desktops, etc.) can have a direct impact on the servers network.

NeMIS Infrastructure: Storage

9 servers used as storage area network (SAN)

- block devices exported by ATA over Ethernet (AoE)
- some block devices locally redundant using software raid
- some block devices have no redundancy

NeMIS Infrastructure: Virtualization servers

27 servers that host virtual machines

- Circa 330 `Xen PV VMs`
- Most of the `VMs` run on the newer and bigger servers.
- The older servers don't support hardware virtualization

View of our VM services

View of our VM services

The `Xen Map` is an essential tool that gives a view of the `VMs` distribution on the hypervisor servers. We can also see what `VMs` are not running, the uptime, RAM, cpu and other parameters. Because it's completely passive, it's mostly used to find on which hypervisor a `VM` is running or to decide where to create a new `VM`.

Some history

- The infrastructure has grown very rapidly and without time for planning.
- Very basic support tools

Some history

All the most important operations are completely manual

1. Create and export a block device on a SAN
2. Create a virtual machine
3. Attach a SAN block device to a virtual machine
4. Move a virtual machine to another host
5. Install and configure the needed software components

Some history: monitoring

We had some monitoring tools

- `Munin` as the only global monitoring system. Without any local configuration, provides systems metrics. From `hypervisors`, `SAN` servers, `VMs`
- `Nagios` monitoring for most of the D4Science services (located at CERN until a year ago, now hosted here)

Some history: backups

Centralized backups for servers and some desktops.
Performed by `BackupPC`

Network

We are trying to improve our networking infrastructure

- Slowly working to separate NeMIS servers traffic from the global ISTI one: if possible, we will subnet
- The external link will be upgraded to 1Gb/s in the near future. We just ordered better switches
- We are doubling the bandwidth between storage and hypervisors. More and better switches

Monitoring

We are extending the `Nagios` monitoring coverage

- All the `Dom0` and `SAN` hardware servers
- Some of our infrastructure services
- The `Hadoop` cluster
- Some of the D-Net services
- More D4science services

Monitoring

New monitoring tools have been deployed

- `Ganglia`, to aggregate metrics from host clusters. Used to monitor the D4science services, the `Hadoop` cluster, the `Xen` and `SAN` servers
- `Elasticsearch/Logstash/Kibana (ELK)`. Aggregates logs and displays graphs about them. Used by the `Hadoop` cluster

Provisioning

Some configuration steps are now automated, with the help of a provisioning tool

- Basic `VM, Dom0, SAN` servers configuration
- `Nagios` configuration (not the d4science one)
- `Ganglia` deployment and configuration
- `Hadoop` cluster, `Solr`, `Jenkins` cluster
- Most of the D-Net infrastructure services
- Some small parts of the D4Science services

Medium/Long term goals

- Move away from `AoE` to a distributed file system
- Move away from `Xen PV` to `KVM` and *containers*
- Better monitoring tools
- Cloud management systems
- Automatically provision all the infrastructure

Distributed file system

Ceph `http://ceph.com/` seems to be the best candidate

- Completely distributed
- Object storage, `S3` API compatible
- Block device `(RADOS)`. Striped, replicated. `KVM` can boot from those block devices
- `OpenStack` supports it
- `Ganeti` supports it (via `libvirt`)
- Free software

Distributed file system

Ceph http://ceph.com/ seems to be the best candidate

- Completely distributed
- Object storage, S3 API compatible
- Block device (RADOS). Striped, replicated. KVM can boot from those block devices
- OpenStack supports it
- Ganeti supports it (via libvirt)
- Free software

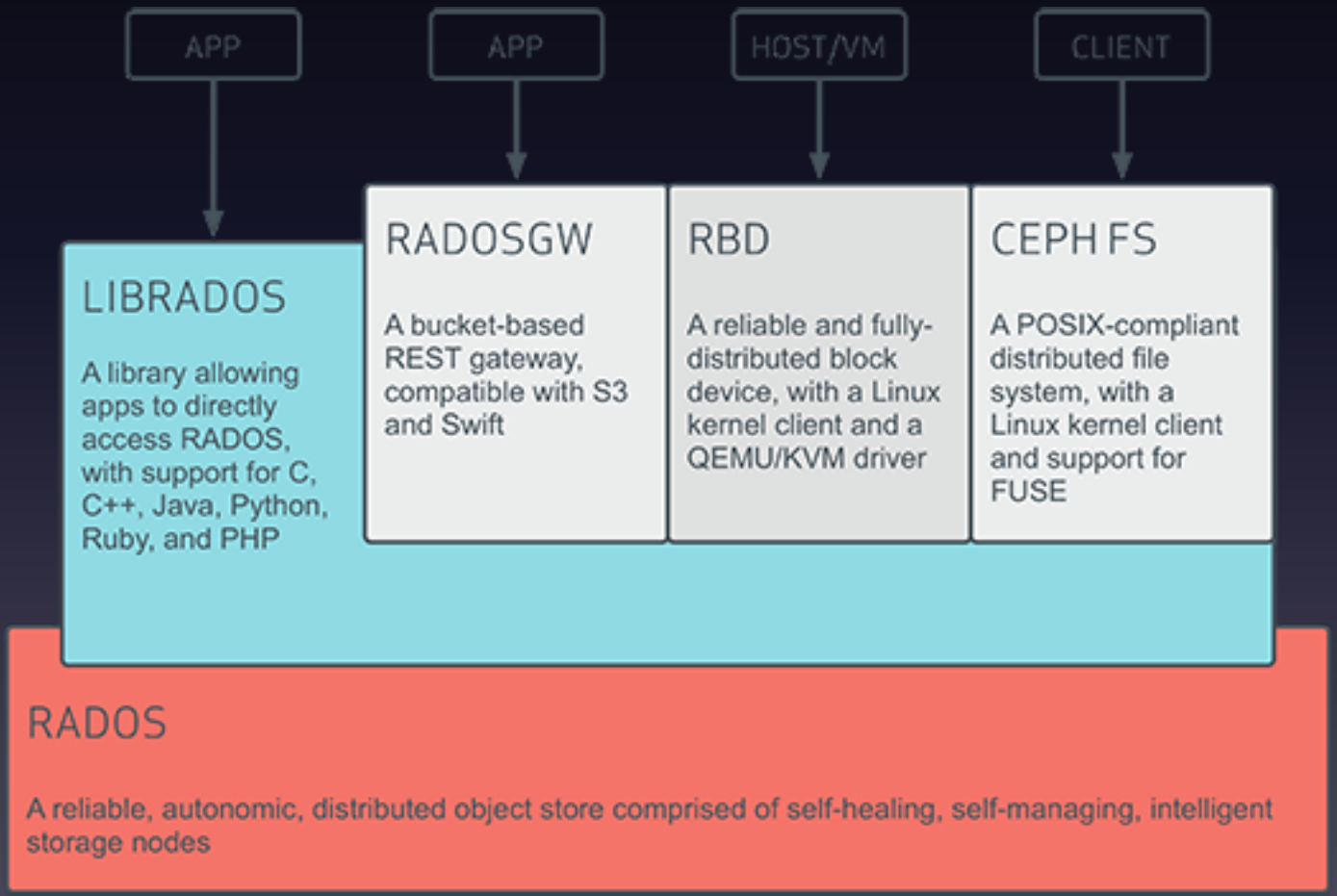
AoE is easy to configure but it has many drawbacks.

Ceph also provides also a POSIX compliant file system. That file system has a linux kernel module and can also be NFS exported. The Ceph authors say that the POSIX file system is not production ready yet.

Ceph architecture

The image is taken from here:

http://docs.ceph.com/docs/master/architecture/



Better virtualization tools. From Xen PV to KVM and containers I

- Xen networking is very fragile
- Xen has big unsolved problems with balooning
- Most of the modern tools have better support for KVM
- Stick with Xen for the Ganeti cluster that uses the older hardware

Better virtualization tools. From Xen PV to KVM and containers I

- Xen networking is very fragile
- Xen has big unsolved problems with balooning
- Most of the modern tools have better support for KVM
- Stick with Xen for the Ganeti cluster that uses the older hardware

We are only using the basic Xen networking features right now so we aren't really suffering of any significant problems. The performance ones are probably related to the fact that we are not using the hardware virtualization support. But we could face problems in the future while trying to move to OpenStack.

Sticking to Xen in the Ganeti cluster permits to use the old servers that do not have hardware virtualization. Ganeti has support for Ceph, that permits to use Ceph block devices as external storage.

Better virtualization tools. From Xen PV to KVM and containers II

We need a much more powerful and flexible environment

- Dynamic provisioning of VMs or containers
- Give some users the possibility to create/destroy VMs
- Integration with the block storage
- Possibility to migrate VMs and containers

Better virtualization tools. From Xen PV to KVM and containers II

We need a much more powerful and flexible environment

- Dynamic provisioning of VMs or containers
- Give some users the possibility to create/destroy VMs
- Integration with the block storage
- Possibility to migrate VMs and containers

Offering a 99.7 availability rate, as documented by Nagios for the D4Science infrastructure was a very hard task considering that:

- Moving VMs across hypervisors require a downtime and a lot of manual work
- Upgrades and maintenance of the hypervisor is an hard task that implies manually moving all the hosted VMs on another hypervisor
- An hardware fault on the hypervisor implies a downtime of all the VMs hosted on that server

Dynamic provisioning of VMs or containers

Cloud manager. OpenStack: http://www.openstack.org/

- Manages VMs, storage, network
- Dashboard to monitor all the activities
- APIs and command line utilities
- High Availability
- Free software

Dynamic provisioning of VMs or containers

Cloud manager. OpenStack: http://www.openstack.org/

- Manages VMs, storage, network
- Dashboard to monitor all the activities
- APIs and command line utilities
- High Availability
- Free software

OpenStack mimicks the behaviour of Amazon AWS, even some of the APIs are similar. It offers the ability to create VM templates and from them VMs with or without permanent storage, VMs migration.

Self service VMs

Openstack manager, ManageIQ:
http://www.manageiq.org/

The final goal is to have a *self service* infrastructure.

- Create abstractions over OpenStack
- Different privileges for different users
- Second level admins
- Adds quota and partitioning abilities to OpenStack
- Users can create instances and clusters of instances within their quota limits without the need of any administrators actions
- Free software

Self service VMs

Openstack manager, ManageIQ:
http://www.manageiq.org/

The final goal is to have a *self service* infrastructure.

- Create abstractions over OpenStack
- Different privileges for different users
- Second level admins
- Adds quota and partitioning abilities to OpenStack
- Users can create instances and clusters of instances within their quota limits without the need of any administrators actions
- Free software

ManageIQ is the base for the Red Hat CloudForms product. It can also manage oVirt and VMWare vSphere installations.
It adds some capabilities that aren't present in the OpenStack dashboard:

- A graphical interface to create VMs
- Quota system, per user
- Users with the ability to create/destroy VMs without touching any other configuration aspect
- Differentiation between development, test and production infrastructures

Better virtualization tools. From `Xen PV` to `KVM` and containers III

Containers manager, Apache Mesos:
`http://mesos.apache.org/`

- `CPU`, memory, storage abstraction
- Supports `docker` and `rocket` containers
- Free software

Better virtualization tools. From `Xen PV` to `KVM` and containers IV

Containers orchestration, Kubernetes:
`http://kubernetes.io/`

- Supports `docker`, not `rocket` (yet?)
- Manages workloads
- Groups containers that are part of the same application
- Free software

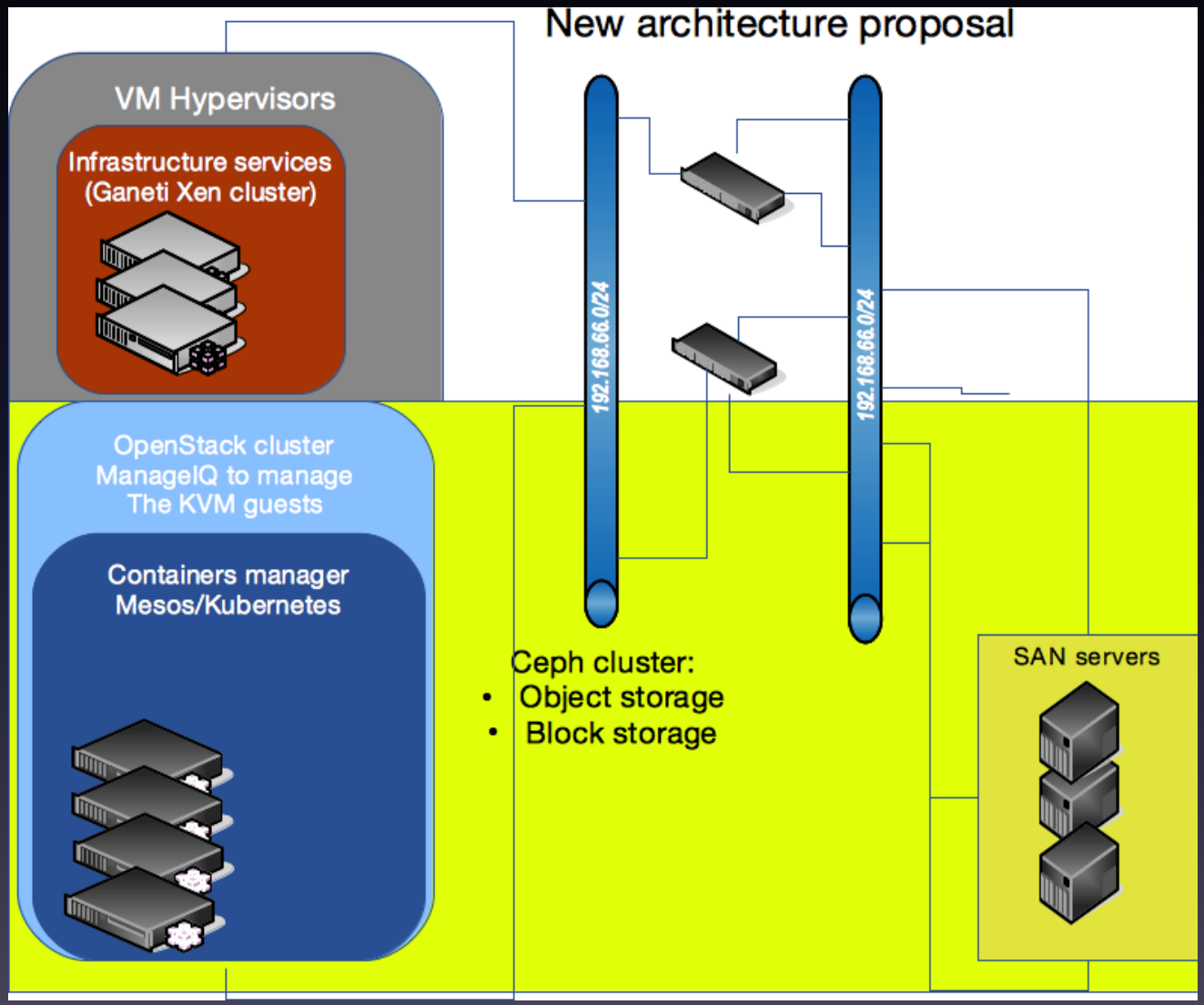
Better virtualization tools. From Xen PV to KVM and containers IV

Containers orchestration, Kubernetes:
http://kubernetes.io/

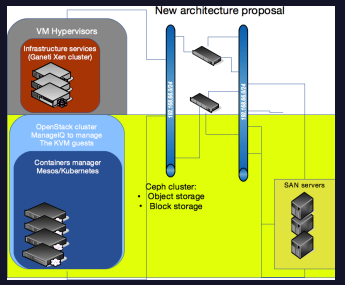
- Supports docker, not rocket (yet?)
- Manages workloads
- Groups containers that are part of the same application
- Free software

Mesos, Kubernetes and containers in general will be an argument of a Marko's speech.

New architecture proposal



New architecture proposal



The `ganeti` cluster can be extended to include the older hardware while newer ones come in.

The ceph cluster can include the newer hypervisor nodes too, so that we can exploit the local disks and still have the ability to migrate `VMs` and containers to a different hypervisor.

New architecture, a schedule?

1. Start with a `Ceph` cluster on some new disks installed on the available storage servers
2. Experiment `Openstack` on a new server
3. Install `ManageIQ` on top of `OpenStack`
4. Start the `VM` migration and deploy `OpenStack` on the available hardware
5. Install `Mesos/Kubernetes` on top of `OpenStack` and start playing with containers (`docker`, `rocket`)

New architecture, a schedule?

1. Start with a `Ceph` cluster on some new disks installed on the available storage servers
2. Experiment `Openstack` on a new server
3. Install `ManageIQ` on top of `OpenStack`
4. Start the `VM` migration and deploy `OpenStack` on the available hardware
5. Install `Mesos/Kubernetes` on top of `OpenStack` and start playing with containers (`docker`, `rocket`)

We can use the available storage dedicated servers, with additional disks, to start deploying `Ceph`.

A new server is needed to start working on `OpenStack`.

A couple of 48 ports 10Gb/s switches could permit us to dismiss the old external switches and have a much better interconnection network. We need some 10Gb/s 2/4 ports ethernet cards too.

Better monitoring tools

- Get rid of `munin` (and maybe `ganglia`?)
- Better `nagios` instrumentation
  - metrics
  - more effective checks
  - better GUI?
  - collect alarms from the other monitoring tools
- New tools to collect metrics (`prometheus`?)
- Get rid of local logs when possible (send them to `ELK`?)

Better monitoring tools

- Get rid of `munin` (and maybe `ganglia`?)
- Better `nagios` instrumentation
  - metrics
  - more effective checks
  - better GUI?
  - collect alarms from the other monitoring tools
- New tools to collect metrics (`prometheus`?)
- Get rid of local logs when possible (send them to `ELK`?)

`prometheus` seems very promising and some of its features are much more advanced than the equivalent from ganglia or ELK. I fear that we will not able to use it exclusively, at least on the short term. And we will need - a big - help from the applications, to collect and manipulate their metrics

Provisioning. Why?

- Tasks automation (configuration, rolling upgrades, etc.)
- Keep track of systems and apps configurations
- Self document the infrastructure
- Time saving
- Delegate operations

Provisioning tools

Lots of available choices

- cfengine: http://cfengine.com
- Chef: https://www.chef.io
- Puppet: https://puppetlabs.com
- Capistrano: http://stackshare.io/capistrano
- Ansible: http://www.ansible.com/

Provisioning tools comparison

	ansible	capistr.	cfengine	chef	puppet
lang	YAML	ruby	propriet.	ruby	ruby
mode	push (ssh)	push (ssh)	pull (central server)	pull (central server)	pull (central server)
main role	provision configure orchestrate	provision	configure	configure provision	configure provision

Provisioning tools comparison

	ansible	capistr.	cfengine	chef	puppet
lang	YAML	ruby	propriet.	ruby	ruby
mode	push (ssh)	push (ssh)	pull (central server)	pull (central server)	pull (central server)
main role	provision configure orchestrate	provision	configure	configure provision	configure provision

Most of the tools other than `ansible` are good at provisioning/configuring or orchestration only.

The `puppet` users for example usually use `facter` as a primitive orchestration tool, and `capistrano` is used in conjunction with `chef` or `puppet`

Ansible I

Why `ansible`

- Powerful but with a very simple syntax
- Connects to target hosts via `ssh`, no additional software required to start using it
- Can switch user via `sudo` if needed
- Lot of modules that simplify its use
- Can be used to execute single commands, as a more sophisticated *distributed* `ssh`

Ansible II

- No central server means that can be run from any desktop computer
- Push based
- Can be used for *configuration, deployment, orchestration*
- The latest versions introduce a pull mode

Ansible III

Ansible book is a soon to be published book. The introductory chapters are free to download and are a good help for the beginners. More free documentation can be found on the `ansible` docs site

`Ansible` terminology

`Inventory` Description of the target hosts (list or groups of hostnames. Optionally variables can be associated)

`Variable` Used inside tasks and templates

`Task` Basic `ansible` action

`Role` A playbook that can be included in more than one play

`Playbook` Inventory + variables + tasks (+ roles)

Inventory

- Hosts
- Host groups
- Eventually, variables
- Inventory can be dynamic
- Mixing static files and dynamic inventory is possible

Variables

- YAML syntax
- Booleans
- Strings
- Lists
- Dictionaries

Tasks I

- A task executes a single action. An action is a module. We can associate a description, use variables and conditionals
- Modules are idempotent
- Tasks are executed in order, one at a time
- Tasks are executed in the same way on all the hosts involved
- When a task fails on a host, that host is excluded by the execution
- Handlers by default are not run on failed hosts

Tasks II

A small example:

Listing 1: First task example

```
- name: install python-apt
  raw: "apt-get update; apt-get install -y
     python-apt lsb-release"
  when: has_apt
  tags:
    - pythonapt
```

Tasks II

A small example:

Listing 1: First task example

```
- name: install python-apt
  raw: "apt-get update; apt-get install -y
     python-apt lsb-release"
  when: has_apt
  tags:
    - pythonapt
```

This task doesn't use any of the `ansible` `python` requirements. Can be used to install the minimal environment that permits to run the ansible modules on the target host

Tasks III

In the earlier example, `raw` was the ansible module that we run. A list of the available modules can be obtained running the command

```
ansible-doc -l
```

While the documentation of each module is available running the command

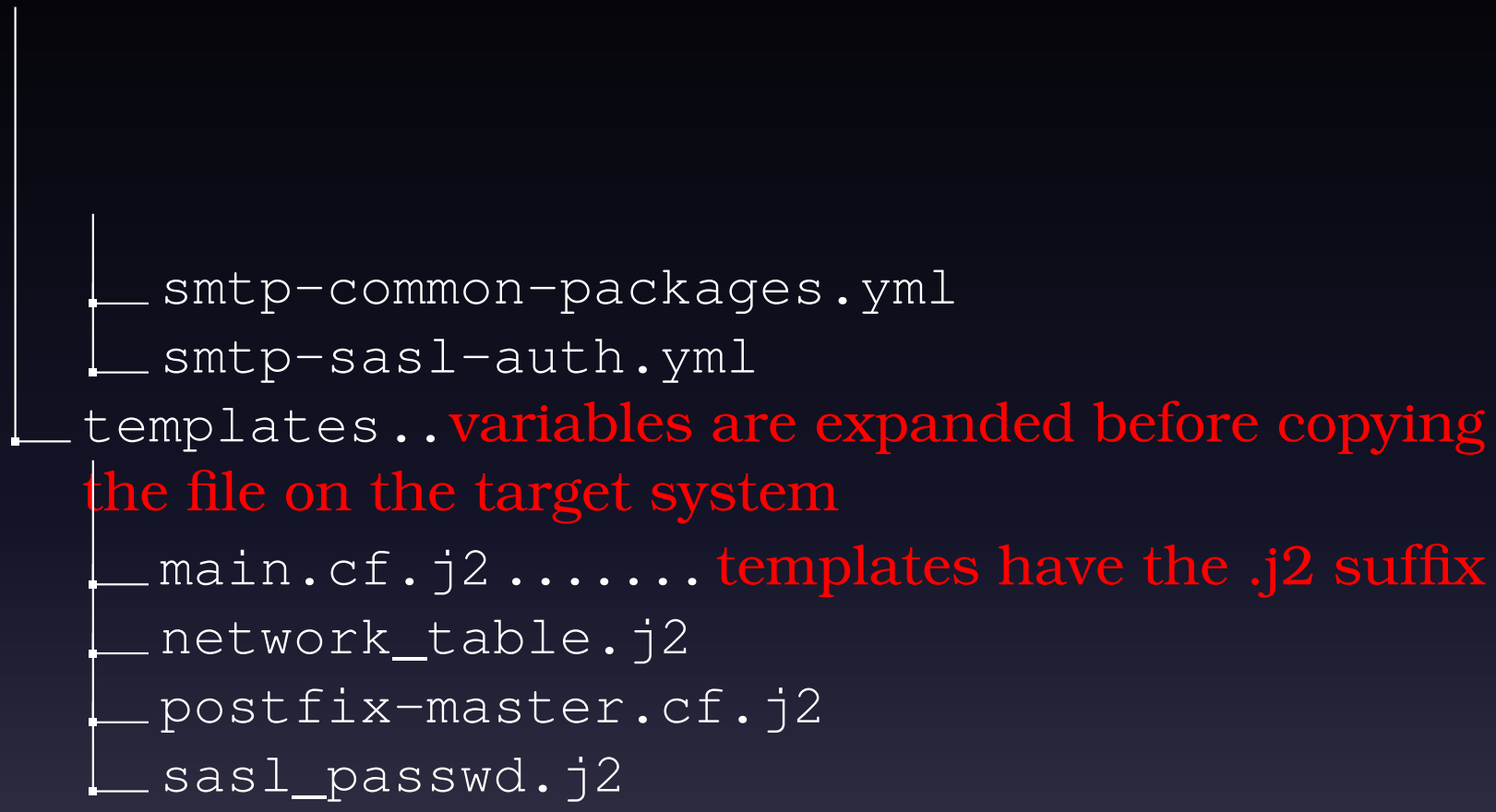
```
ansible-doc modulename
```

Roles I

A role is a collection of one or more tasks, with its variables and eventually templates and files. A role structure:

```
/postfix-relay/ ..... root directory
├── defaults/
│   └── main.yml ..... default variables go here
├── files/ .. files that are copied without modifications
│   └── sasl_smtpd.conf
├── handlers/
│   └── main.yml ..... start/stop service, whatever
│                      conditional action
└── tasks/
    ├── main.yml . put your tasks here, or include other
    │              files
    └── postfix-relay-server.yml
```

Roles II

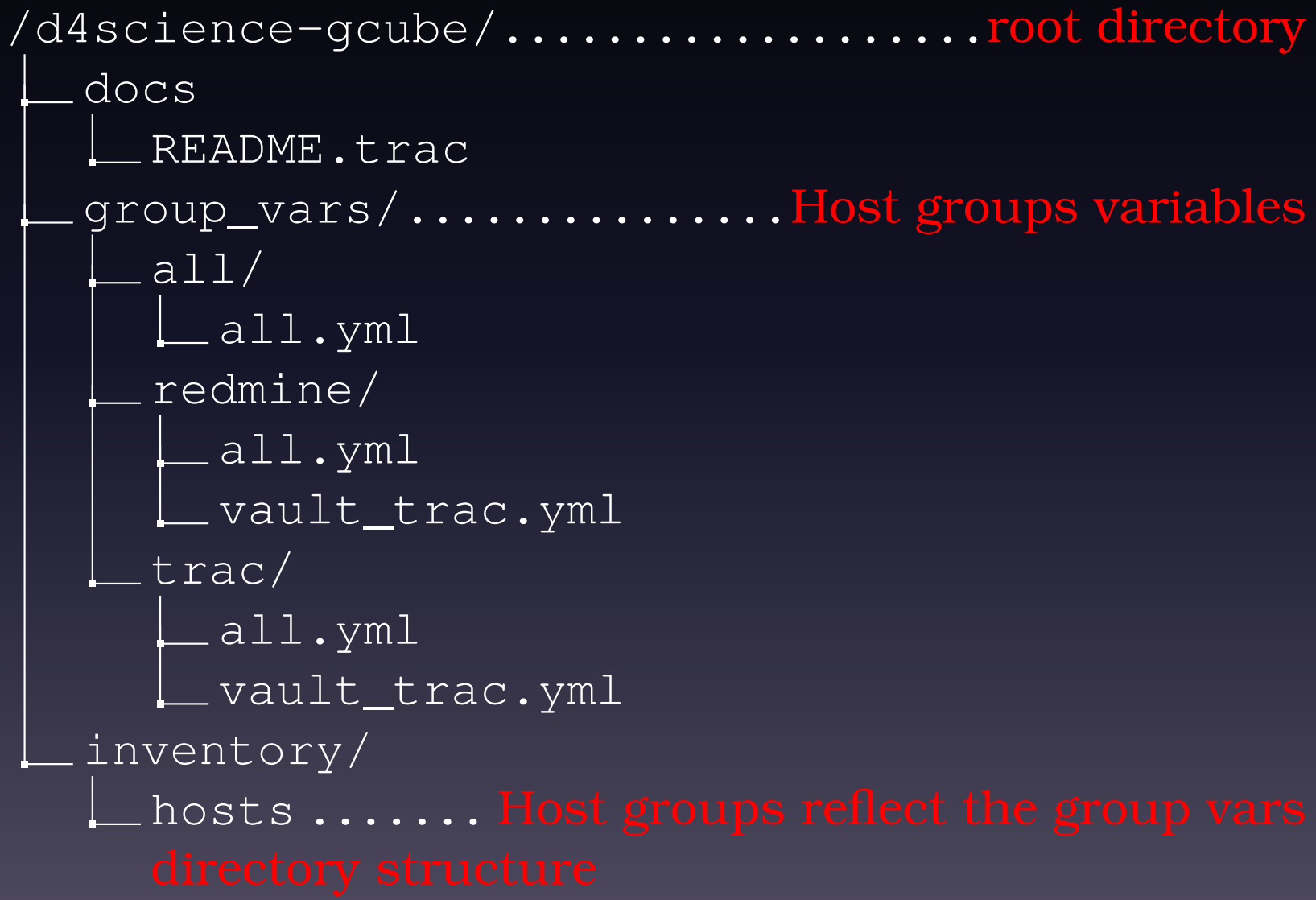

smtp-common-packages.yml
smtp-sasl-auth.yml
templates..variables are expanded before copying
the file on the target system
main.cf.j2 ..... templates have the .j2 suffix
network_table.j2
postfix-master.cf.j2
sasl_passwd.j2

Playbooks

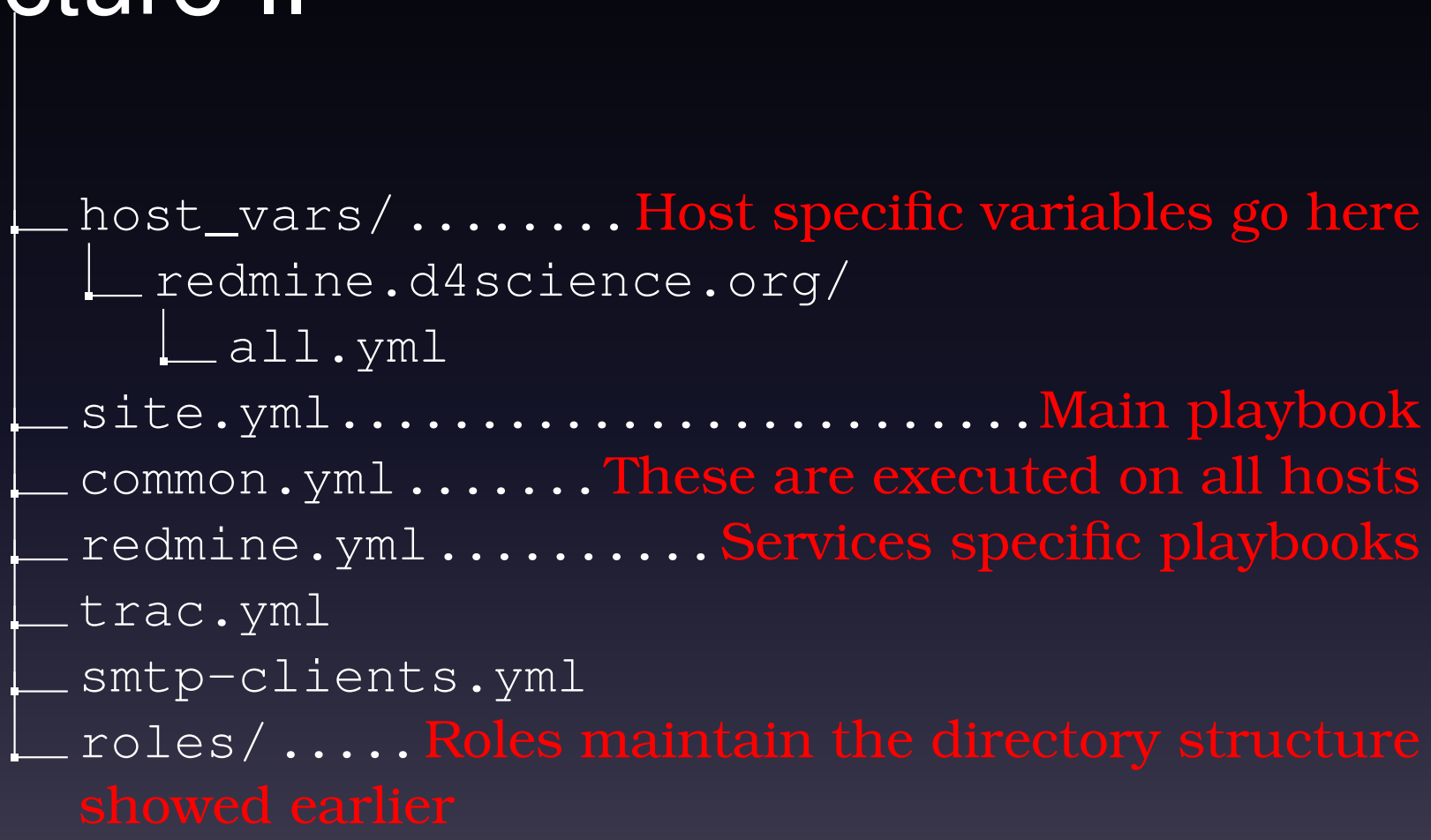
A configuration management script

- inventory + variables + tasks (+ roles)
- It runs and executes tasks

Playbooks best practices: directory structure I



Playbooks best practices: directory structure II



Playbooks best practices: directory structure II

```
host_vars/.....Host specific variables go here
  redmine.d4science.org/
    all.yml
site.yml.....Main playbook
common.yml.....These are executed on all hosts
redmine.yml.....Services specific playbooks
trac.yml
smtp-clients.yml
roles/.....Roles maintain the directory structure
showed earlier
```

The main playbook includes all the other ones. We use it when we want to run the complete playbook.

The host groups are defined inside the inventory

Example: inventory

From the `social-isti` playbook that manages `nagios` and `ganglia`

Listing 2: Inventory example

```
[social_isti:children]
cassandra_si
gcore_ghn_si

[cassandra_si]
cassandra1-si.isti.cnr.it
cassandra2-si.isti.cnr.it

[gcore_ghn_si]
node[1:8]-si.isti.cnr.it
```

Example: inventory

From the `social-isti` playbook that manages `nagios` and `ganglia`

Listing 2: Inventory example

```
[social_isti:children]
cassandra_si
gcore_ghn_si

[cassandra_si]
cassandra1-si.isti.cnr.it
cassandra2-si.isti.cnr.it

[gcore_ghn_si]
node[1:8]-si.isti.cnr.it
```

Hostnames can be aggregated using a regexp like syntax. And can be grouped. The inventory can store variables, too

Example: Variables I

From the `social-isti` playbook that manages `nagios` and `ganglia`

Listing 3: Variables example

```
social_isti_db_name: social-isti-db

psql_db_data:
  - { name: '{{ social_isti_db_name }}',
      encoding: 'UTF8', user: 'postgres', pwd:
      '', roles: '', allowed_hosts: [ '{{
      network.isti }}/32' ] }

si_ghn_gcube_port: 8080
si_jackrabbit_port: 9000
si_tomcat_port: 9090
si_cassandra_jmx_port: 7199
```

Example: Variables II

From the `social-isti` playbook that manages `nagios` and `ganglia`

Listing 4: Variables example

```
iptables_default_policy: REJECT
iptables:
  tcp_rules: True
  tcp:
    - { port: '1024:65535', allowed_hosts: [
        '146.48.85.155', '146.48.85.156',
        '146.48.85.157' ] }
    - { port: '1024:65535', allowed_hosts: [
        '146.48.87.174', '146.48.87.112', '{{
        nagios_monitoring_server_ip }}' ] }
```

Example: Variables III

Listing 5: Variables example

```
nagios_contactgroup: si-managers
nagios_service_contacts:
  - { name: 'lino', email: 'pasquale.
      pagano@isti.cnr.it', submit_commands: '0',
       period: '24,7', create: False }
  - { name: 'roberto', email: 'roberto.
      cirillo@isti.cnr.it', submit_commands:
      '1', period: '24,7', create: False }
nagios_postgresql_check: True
mongodb_nagios_user: administrator
#mongodb_nagios_user_pwd: see the vault file
```

Variables IV: ansible vault

Ansible vault

- Encrypts files
- It's only possible to encrypt yaml files containing variables, for now
- Same password for all the playbook's encrypted files
- Encrypted files can be edited directly with `ansible-vault edit`

Variables IV: ansible vault

Ansible vault

- Encrypts files
- It's only possible to encrypt yaml files containing variables, for now
- Same password for all the playbook's encrypted files
- Encrypted files can be edited directly with `ansible-vault edit`

A *vault file* is an encrypted `yaml` file. The files are automatically unencrypted by ansible while running the playbook. This permits to store passwords and other sensible data into the versioning systems in a safe way. The only drawback is that only one password for playbook is permitted

Variables V

Variables can be defined in different places. Each place means different priorities.
From lower to higher:

- defaults
- group_vars
- vars files inside playbooks
- host_vars
- command line

Example: Tasks I

Listing 6: A task is an action. A small example

```
- name: install python-apt
  raw: "apt-get update; apt-get install -y
     python-apt lsb-release"
  when: has_apt
  tags:
    - pythonapt
```

The `raw` module is special, because it does not require any python presence on the target machine. It can be used to setup the minimal requirements.

Example: Tasks II

Listing 7: Where we use one of the modules

```
- name: Install python-software-properties
  apt: pkg=python-software-properties state=
      installed
  when: has_apt
  tags:
    - pythonapt
```

---

ansible modules are usually idempotent: they do nothing if there's nothing to do.

Example: Tasks II

Listing 7: Where we use one of the modules

```
- name: Install python-software-properties
  apt: pkg=python-software-properties state=
     installed
  when: has_apt
  tags:
    - pythonapt
```

ansible modules are usually idempotent: they do nothing if there's nothing to do.

Conditionals can be used to effectively execute a task only when the condition is satisfied

Example: Tasks III

Listing 8: Were we use variables as conditions

```
- name: Fix rsyslog behaviour on some ubuntu
    machines disabling the kernel logger
  lineinfile: dest=/etc/rsyslog.conf regexp="\\
      $ModLoad\ imklog" line="#$ModLoad imklog"
      backup=yes
  when:
    - is_precise and ansible_kernel !=
        "3.2.0-4-amd64"
    - is_not_trusty
  notify:
    Restart rsyslog
```

Example: Tasks III

Listing 8: Were we use variables as conditions

```
- name: Fix rsyslog behaviour on some ubuntu
    machines disabling the kernel logger
  lineinfile: dest=/etc/rsyslog.conf regexp="\\
      $ModLoad\ imklog" line="#$ModLoad imklog"
      backup=yes
  when:
    - is_precise and ansible_kernel !=
       "3.2.0-4-amd64"
    - is_not_trusty
  notify:
    Restart rsyslog
```

and we execute a handler if the task is executed and the result is that a change happened

Examples: Roles

A role that installs a single `mongodb` instance and limits access by using iptables firewall rules

- Installs the additional repository from mongodb.org
- Installs the packages
- Configure some defaults
- Enable and start the service
- Defines rules used by the iptables role

Role structure

```
/mongodb
├── defaults
│   └── main.yml
├── handlers
│   └── main.yml
├── tasks
│   └── main.yml
└── templates
    └── mongodb-2.4.conf.j2
```

Default variables

```
mongodb_install_from_external_repo: True
mongodb_start_server: 'yes'
mongodb_tcp_port: 27017
mongodb_http_interface: False
mongodb_http_port: 28017
mongodb_user: mongodb
mongodb_group: mongodb
mongodb_logdir: /var/log/mongodb
mongodb_logpath: '{{ mongodb_logdir }}/mongodb.
   log'
mongodb_dbpath: /var/lib/mongodb
mongodb_directoryperdb: False
mongodb_allowed_hosts:
  - '{{ ansible_fqdn }}/32'
  - 127.0.0.1/8
```

Template details

```
...
dbpath={{ mongodb_dbpath }}
directoryperdb={{ mongodb_directoryperdb }}
port = {{ mongodb_tcp_port }}
{% if not mongodb_http_interface %}
# Disable the HTTP interface (Defaults to
    localhost:28017).
nohttpinterface = true
{% endif %}
...
```

Handlers

```
---
- name: Update apt cache
  apt: update_cache=yes
  ignore_errors: true

- name: Restart mongodb
  service: name=mongodb state=restarted
```

Tasks detail: configure the mongodb repository

```
- name: Install the mongodb apt key
  raw: apt-key adv --keyserver hkp://keyserver.
      ubuntu.com:80 --recv 7F0CEB10
  when: mongodb_install_from_external_repo
  tags: mongodb

- name: Install the mongodb repository
  copy: content="deb http://downloads-distro.
      mongodb.org/repo/ubuntu-upstart dist 10gen
      " dest=/etc/apt/sources.list.d/mongodb.
      list owner=root group=root mode=044
  when: mongodb_install_from_external_repo
  register: external_repo
  tags: mongodb

- name: Update the apt cache
  apt: update_cache=yes
  when: ( external_repo | changed )
  tags: mongodb
```

Tasks detail: configure the mongodb repository

```
- name: Install the mongodb apt key
  raw: apt-key adv --keyserver hkp://keyserver.
     ubuntu.com:80 --recv 7F0CEB10
  when: mongodb_install_from_external_repo
  tags: mongodb

- name: Install the mongodb repository
  copy: content="deb http://downloads-distro.
     mongodb.org/repo/ubuntu-upstart dist 10gen
     " dest=/etc/apt/sources.list.d/mongodb.
     list owner=root group=root mode=044
  when: mongodb_install_from_external_repo
  register: external_repo
  tags: mongodb

- name: Update the apt cache
  apt: update_cache=yes
  when: ( external_repo | changed )
  tags: mongodb
```

Here we configure the mongodb repository, then we update the package manager cache

Task detail: install the mongodb package

```
- name: Install the mongodb server from the
    external repo
  apt: pkg={{ item }} state=installed
  with_items:
    - mongodb-10gen
  when: mongodb_install_from_external_repo
  tags: mongodb

- name: Install the mongodb server from the
    distribution repo
  apt: pkg={{ item }} state=installed
  with_items:
    - mongodb-server
  when: not mongodb_install_from_external_repo
  tags: mongodb
```

Task detail: install the mongodb package

```
- name: Install the mongodb server from the
    external repo
  apt: pkg={{ item }} state=installed
  with_items:
    - mongodb-10gen
  when: mongodb_install_from_external_repo
  tags: mongodb

- name: Install the mongodb server from the
    distribution repo
  apt: pkg={{ item }} state=installed
  with_items:
    - mongodb-server
  when: not mongodb_install_from_external_repo
  tags: mongodb
```

Here we install the mongodb server package. The one that comes with the distribution or the one from the mongodb repository

Task detail: configure mongodb

```
- name: Install the mongodb defaults file
  copy: content="ENABLE_MONGODB={{
      mongodb_start_server }}" dest=/etc/default
      /mongodb owner=root group=root mode=0444
  tags: mongodb
- name: Create the mongodb db directory
  file: dest={{ mongodb_dbpath }} state=
      directory owner={{ mongodb_user }} group
      ={{ mongodb_group }} mode=0755
  tags: mongodb
- name: Create the mongodb log directory
  file: dest={{ mongodb_logdir }} state=
      directory owner={{ mongodb_user }} group
      ={{ mongodb_group }} mode=0755
  tags: mongodb
- name: Install the mongodb 2.4 configuration
  template: src=mongodb-2.4.conf.j2 dest=/etc/
      mongodb.conf owner=root group=root mode
      =0444
  when: ( mongodb_start_server is defined ) and
       ( mongodb_start_server == 'yes' )
  notify: Restart mongodb
```

Task detail: set the service status

```
- name: Ensure mongodb is started
  service: name=mongodb state=started enabled=
      yes
  when: ( mongodb_start_server is defined ) and
       ( mongodb_start_server == 'yes' )
  tags: mongodb

- name: Ensure mongodb is stopped and disabled
  service: name=mongodb state=stopped enabled=
      no
  when: ( mongodb_start_server is defined ) and
       ( mongodb_start_server == 'no' )
  tags: mongodb
```

Task detail: set the service status

```
- name: Ensure mongodb is started
  service: name=mongodb state=started enabled=
     yes
  when: ( mongodb_start_server is defined ) and
      ( mongodb_start_server == 'yes' )
  tags: mongodb

- name: Ensure mongodb is stopped and disabled
  service: name=mongodb state=stopped enabled=
     no
  when: ( mongodb_start_server is defined ) and
      ( mongodb_start_server == 'no' )
  tags: mongodb
```

We use the mongodb_start_server variable to choose if the service is to be enabled or disabled

Task detail: where the mongodb access is regulated by the iptables role

```
{% if mongodb_allowed_hosts is defined %}
# mongodb clients
{% for ip in mongodb_allowed_hosts %}
-A INPUT -m state --state NEW -s {{ ip }} -p
    tcp -m tcp --dport {{ mongodb_tcp_port }} -j
     ACCEPT
{% endfor %}
-A INPUT -p tcp -m tcp --dport {{
    mongodb_tcp_port }} -j DROP
{% endif %}
```

Task detail: where the mongodb access is regulated by the iptables role

```
{% if mongodb_allowed_hosts is defined %}
# mongodb clients
{% for ip in mongodb_allowed_hosts %}
-A INPUT -m state --state NEW -s {{ ip }} -p
   tcp -m tcp --dport {{ mongodb_tcp_port }} -j
    ACCEPT
{% endfor %}
-A INPUT -p tcp -m tcp --dport {{
   mongodb_tcp_port }} -j DROP
{% endif %}
```

When we run the iptables role in a playbook that installs mongodb, some mongodb specific rules are enabled

Playbook detail: where iptables and mongodb are used together

The file name is mongodb-servers.yml

```
---
- hosts: mongo-servers
  remote_user: root
  vars_files:
    - ../library/vars/isti-global.yml
  roles:
    - ../library/ubuntu-deb-general
    - ../library/iptables
    - ../library/ssh-keys
    - ../library/mongodb
```

Playbook detail: where iptables and mongodb are used together

The file name is mongodb-servers.yml

```
---
- hosts: mongo-servers
  remote_user: root
  vars_files:
    - ../library/vars/isti-global.yml
  roles:
    - ../library/ubuntu-deb-general
    - ../library/iptables
    - ../library/ssh-keys
    - ../library/mongodb
```

The *library* roles are custom made roles that are designed to be used as playbook building blocks, respecting the needs of the NeMIS laboratory architecture

Playbook: how to play

Where we do something, finally

```
user@machine> ansible-playbook mongodb-servers.
    yml -i inventory/hosts
```

- `ansible-playbook` is the command that executes the playbook
- `mongodb-servers.yml` is the playbook file
- `inventory/hosts` is the inventory file

Playbook output I

```
PLAY [mongo-servers] ***********************************************************

GATHERING FACTS ****************************************************************
ok: [ubuntu]

TASK: [../../library/timezone | Write the timezone file] ***********************
ok: [ubuntu]

TASK: [../../library/timezone | Reconfigure the system tzdata] ****************
skipping: [ubuntu]

TASK: [../../library/deb-set-locale | Generate locales] ************************
ok: [ubuntu]

TASK: [../../library/deb-set-locale | Update the locale default] ***************
ok: [ubuntu]

TASK: [../../library/iptables | Install the needed iptables packages] *********
ok: [ubuntu] => (item=iptables,iptables-persistent)

TASK: [../../library/iptables | Install the IPv4 rules with a different name. Needed by Ubuntu < 12.04] ***
skipping: [ubuntu] => (item=rules.v4)

TASK: [../../library/iptables | Install the IPv4 and IPv6 iptables rules. The IPv6 ones are not used] ***
ok: [ubuntu] => (item=rules.v4)
ok: [ubuntu] => (item=rules.v6)

TASK: [../../library/fail2ban | install fail2ban ubuntu >= 14.04] *************
ok: [ubuntu] => (item=fail2ban)

TASK: [../../library/fail2ban | Install the fail2ban custom jail file] ********
changed: [ubuntu]

TASK: [../library/ubuntu-deb-general | install python-apt] *********************
ok: [ubuntu]

TASK: [../library/ubuntu-deb-general | Install python-software-properties] ****
ok: [ubuntu]

TASK: [../library/ubuntu-deb-general | Install software-properties-common on quantal distributions] ***
skipping: [ubuntu]
```

The play starts on the selected hosts or hostgroup. Before actually running the first task, information from all hosts are gathered

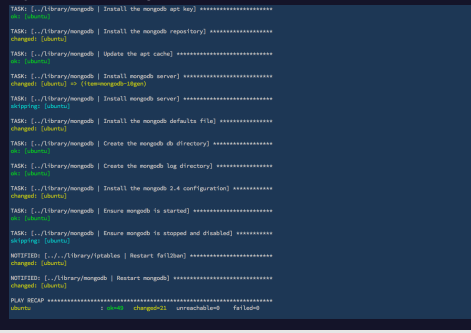
Playbook output 2

```
TASK: [../library/ubuntu-deb-general | setup system apt repository] ***********
changed: [ubuntu]

TASK: [../library/ubuntu-deb-general | Update the apt cache] *****************
ok: [ubuntu]

TASK: [../library/ubuntu-deb-general | install common packages] ***************
changed: [ubuntu] => (item=zile,dstat,iotop,wget,vim-tiny,psmisc,tcpdump,lsof,strace,rsync,multitail,unzip,htop,tree,bind9-host,bash-comple

TASK: [../library/ubuntu-deb-general | Install the ntp server] ***************
changed: [ubuntu]

TASK: [../library/ubuntu-deb-general | Ensure that the ntp server is running] ***
ok: [ubuntu]

TASK: [../library/ubuntu-deb-general | Remove unneeded base packages] *********
ok: [ubuntu] => (item=True)

TASK: [../library/ubuntu-deb-general | Remove unneeded X packages] ************
skipping: [ubuntu]

TASK: [../library/ubuntu-deb-general | Remove the nfs packages] ***************
skipping: [ubuntu]

TASK: [../library/ubuntu-deb-general | Remove rpcbind packages] ***************
skipping: [ubuntu]

TASK: [../library/ubuntu-deb-general | Ensure that the /etc/sysctl.d directory exists] ***
ok: [ubuntu]

TASK: [../library/ubuntu-deb-general | Disable the in kernel ipv6 support] ****
changed: [ubuntu] => (item=net.ipv6.conf.all.disable_ipv6)
changed: [ubuntu] => (item=net.ipv6.conf.default.disable_ipv6)
changed: [ubuntu] => (item=net.ipv6.conf.lo.disable_ipv6)

TASK: [../library/ubuntu-deb-general | file dest=/etc/modprobe.d/00-ipv6-disable.conf state=absent] ***
skipping: [ubuntu]

TASK: [../library/ubuntu-deb-general | file dest=/etc/modutils/disable-ipv6 state=absent] ***
skipping: [ubuntu]

TASK: [../library/ubuntu-deb-general | file dest=/etc/sysctl.d/10-ipv6-disable.conf state=absent] ***
skipping: [ubuntu]
```

Playbook output 3

```
TASK: [../library/mongodb | Install the mongodb apt key] **********************
ok: [ubuntu]

TASK: [../library/mongodb | Install the mongodb repository] *******************
changed: [ubuntu]

TASK: [../library/mongodb | Update the apt cache] *****************************
ok: [ubuntu]

TASK: [../library/mongodb | Install mongodb server] ***************************
changed: [ubuntu] => (item=mongodb-10gen)

TASK: [../library/mongodb | Install mongodb server] ***************************
skipping: [ubuntu]

TASK: [../library/mongodb | Install the mongodb defaults file] ****************
changed: [ubuntu]

TASK: [../library/mongodb | Create the mongodb db directory] ******************
ok: [ubuntu]

TASK: [../library/mongodb | Create the mongodb log directory] *****************
ok: [ubuntu]

TASK: [../library/mongodb | Install the mongodb 2.4 configuration] ************
changed: [ubuntu]

TASK: [../library/mongodb | Ensure mongodb is started] ************************
ok: [ubuntu]

TASK: [../library/mongodb | Ensure mongodb is stopped and disabled] ***********
skipping: [ubuntu]

NOTIFIED: [../../library/iptables | Restart fail2ban] *************************
changed: [ubuntu]

NOTIFIED: [../library/mongodb | Restart mongodb] ******************************
changed: [ubuntu]

PLAY RECAP ********************************************************************
ubuntu                     : ok=49   changed=21   unreachable=0    failed=0
```

Playbook output 3



This is the complete `mongodb` role execution. We can see the handlers run at the end of all the tasks, and the `play recap` phase with a synthesis of the results (one line per host)

ansible-playbook options I

`ansible-playbook` accepts options that can be used to limit the execution or change the environment.

The complete options list can be obtained running

```
ansible-playbook -help
```

ansible-playbook options II

`-step` One step at a time. Asks confirmation after each step

`-tags=tag1,tagN` Use tags to limit the execution to the tasks that match that tags

`-skip-tags=tag1,tagN` Skip the tasks that match the tags

`-l inventory_name` Limit the execution to the specified hostname or inventory group

`-extra-vars=` Set variables in key=value format or JSON

ansible-playbook options III

`-list-tags` Lists all the available tags

`-list-hosts` Outputs the list of matching hosts

`-list-tasks` Lists the tasks that would be executed

`-syntax-check` Perform a syntax check of the playbook, without executing it

Orchestration: pre/post tasks and delegation

It is possibile to execute tasks before and after each play

```
pre_tasks:
  - name: disable nagios alerts for this host
      services
    nagios: action=disable_alerts host={{
       inventory_hostname }} services="all"
    delegate_to: "{{ item }}"
    with_items: groups.monitoring
```

This task disable the nagios alerts before the play. A `post_tasks` task will re-enable them at the end

Orchestration: rolling upgrades

Two features that we can exploit when running upgrades

```
- hosts: all
  remote_user: root
  serial: "25%"
  max_fail_percentage: 10
```

`serial` Can be absolute or a percentage. Here we specify how many target hosts we can run in parallel

`max_fail_percentage` If the threshold is reached, all the play is aborted.

How we are using ansible right now

What we actually have under `ansible` control

- A library of 32 *generic* roles that can be composed to build a playbook
- Playbooks to configure `Xen` hosts and `SAN` servers
- Some infrastructure services: `Nagios, Ganglia`, the `SMTP` relay
- Most of the D-Net infrastructure
- The D-Net `Hadoop` and `Solr` clusters
- Some bits of the D4science infrastructure: the `Ganglia` configuration, `Redmine`, the new *geoserver* machines, the new `SmartGears` nodes, the `CouchDB` service

How we could use ansible in the near feature

1. A developer needs a new service or a new resource
2. A system administrator writes a new playbook or extends an existing one
3. The playbook is tested and run
4. Changes and maintainance can be done by the developers themselves (`iptables` rules, new `tomcat` instances, even new servers configuration)

Long term goals

- Provision all the infrastructure via `ansible`
  - *Corporate* services
  - *Infrastructure* services (OpenStack, Ceph)
  - All the production, development and test environments
- When applicable, use `ansible` to create the service containers (no further configuration is done on running containers)

Any questions?