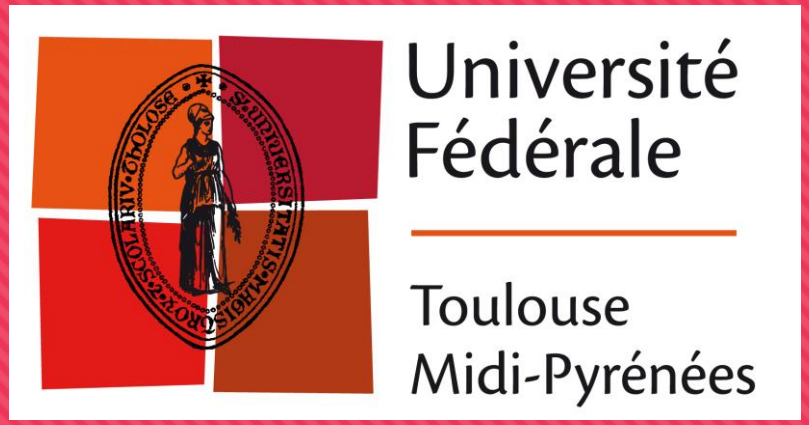


Gestion de parc avec Puppet

Louis Chanouha – 17 octobre 2019 – Capitoul – LAAS-CNRS

Plan

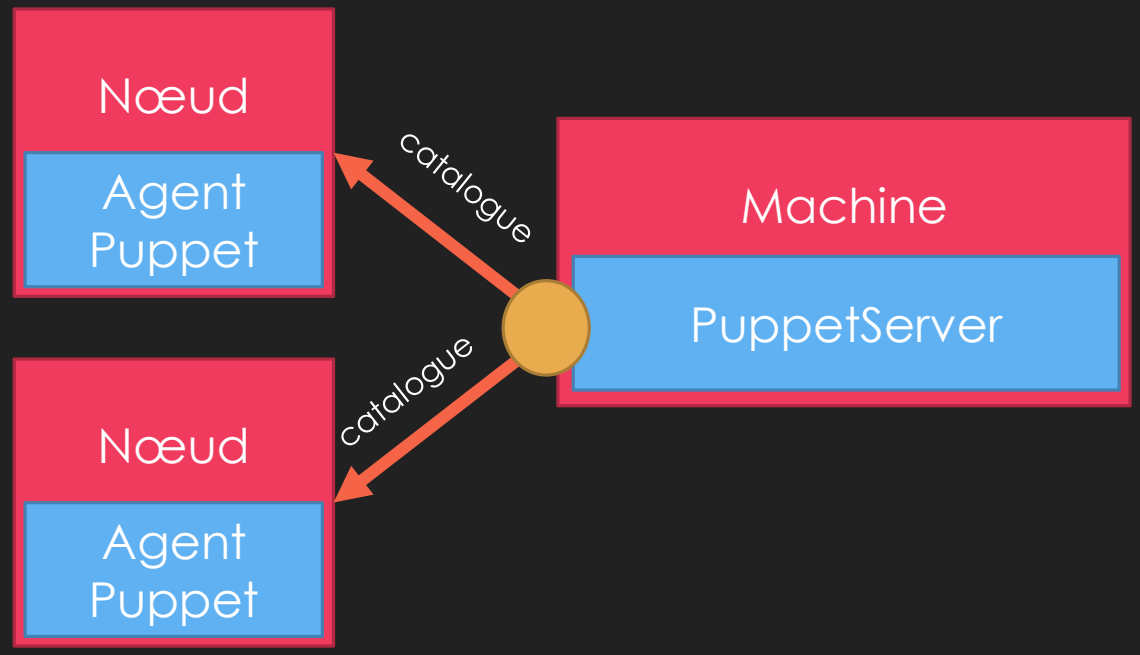
- **Introduction: du besoin à Puppet**
- Architecture de base
- Chronologie d'évolutions
- Focus sur 2 design patterns
- Bilan d'usage

Introduction

- Contexte: nouvelle infrastructure à monter
- Choix de Puppet
  - Capacité d'intégration sur l'existant : s'installe sans conflit sur les machines
  - Cout d'investissement très léger (client => serveur) et mise en place progressive
  - Langage déclaratif mais aussi structuré (conditions, répétitions, modules-classes)
  - Nombreux modules et fonctions de références (on y reviendra)

Architecture de base

- Client(s) - Serveur
  - Un serveur
    - Un démon / port d'écoute unique
    - Un répertoire contenant des configurations sous forme de langage déclaratif
    - Sécurité via un CA
      - Signature des nœuds
  - Un agent à installer sur les VM (« nœuds »)
    - Intégré dans les dépôts Debian à partir de Jessie
    - Exécute le catalogue tous les X temps
      - Démon ou cron (ou les deux)



Architecture de base

- Architecture des fichiers de configuration
  - Un dossier pour chaque environnement : /etc/puppetlabs/code/environments/production/
  - Un sous-dossier manifests
    - Déclaration et configuration des nœuds
  - Un sous-dossier modules
    - /manifests: Écriture de « classes » de nœuds
    - /files & /templates : ressources personnalisées
  - Très facilement versionnable (Git)
    - Attention aux mots de passe !

/manifests/ns1.univ-toulouse.fr.pp

```
node 'ns1.univ-toulouse.fr' {

    # Déclarations spécifiques au serveur ns1

    class { 'ut_debian_base':
        zsh_users => ['root'],
    }

}
```

Plan

- Introduction: du besoin à Puppet
- Architecture de base
- **Chronologie d'évolutions**
- Focus sur 2 design patterns
- Bilan d'usage

Chronologie d'évolutions : bases

- Premier besoin = dépôt de fichiers de base
  - DNS
  - NRPE (Nagios)
  - Dépôt des scripts de monitoring
- Avec application plus fine
  - On peut utiliser Augeas
    - Opérations dans fichiers structurés INI, XML, YML. etc
  - Ou traiter simplement par ligne

```
file {"/etc/resolv.conf":
    owner    => "root",
    group    => "root",
    mode     => "0644",
    content  => file("ut_debian_base/etc/resolv.conf")
}
```

```
augeas { 'puppet.conf agent:defaults':
  changes        => [
    "set /files${pdir}puppet.conf/agent/runinterval 1200"]
}
```

```
file_line { 'sshd password auth‘;
    path    => '/etc/ssh/sshd_config',
    line    => "PasswordAuthentication no",
    match   => '^PasswordAuthentication
}
```

Chronologie d'évolutions: bases

- Utilisation des modules
  - Intégration dans un module
  - Et déclaration du module dans un « nœud »
  - Oui, mais.. C'est pas appliqué..

```
node 'ns1.univ-toulouse.fr' {
    file {"/etc/resolv.conf":[…]}
    file {"/etc/nagios/nrpe.cfg":[…]}
    file {"/opt/comue/bin/":[…]}
}
```

/modules/ut_debian_base/manifests/init.pp

```
class ut_debian_base () {
    file {"/etc/resolv.conf":[…]}
    file {"/etc/nagios/nrpe.cfg":[…]}
    file {"/opt/comue/bin/":[…]}
}
```

/manifests/univ-toulouse.fr.pp

```
node /\.univ-toulouse\.fr$/ {
    require ut_debian_base
}
```

Chronologie d'évolutions: bases

- Premier processus: appliquer les configurations automatiquement
  - Exemple: Nagios / NRPE

```
file {"/etc/nagios/nrpe.cfg":
    content  => file("ut_debian_base/etc/nagios/nrpe.cfg"),
    notify   => Service['nagios-nrpe-server']
}
service { 'nagios-nrpe-server':
    ensure   => 'running',
    enable   => true,
}
```

```
$ puppet agent –t
Notice: /Stage[main]/Ut_debian_base/Concat[/etc/nagios/nrpe.cfg]/File[/etc/nagios/nrpe.cfg]/content: content changed
Notice: /Stage[main]/Ut_debian_base/Service[nagios-nrpe-server]: Triggered 'refresh' from 1 event
Notice: Applied catalog in 4.28 seconds
```

Chronologie d'évolutions: bases

- Premier processus: appliquer les configurations automatiquement
  - Vérifiation des fichiers avant déploiement

```
file {"/etc/haproxy/haproxy-testconf.cfg":
    content  => template('ut_loadbalancer/haproxy.cfg'),
    notify   => Service['haproxy']
}

# Test de la validité du fichier
service { ‘haproxy':
    ensure   => 'running',
    enable   => 'true',
    restart  => '( /usr/sbin/haproxy -c -f /etc/haproxy/haproxy-testconf.cfg && /bin/cp -a /etc/haproxy/haproxy-testconf.cfg
/etc/haproxy/haproxy.cfg && /etc/init.d/haproxy reload) || rm /etc/haproxy/haproxy-testconf.cfg',
}
```

Chronologie d'évolutions: bases

Université Fédérale
Toulouse Midi-Pyrénées

- Deuxième processus: installer des applications/dépendances de base
  - Installation de paquets
  - Installations et configuration de logiciels
    - Base Puppet
    - Via modules contrib
    - Téléchargeables sur Puppet Forge
    - 55 modules officiels / 6000

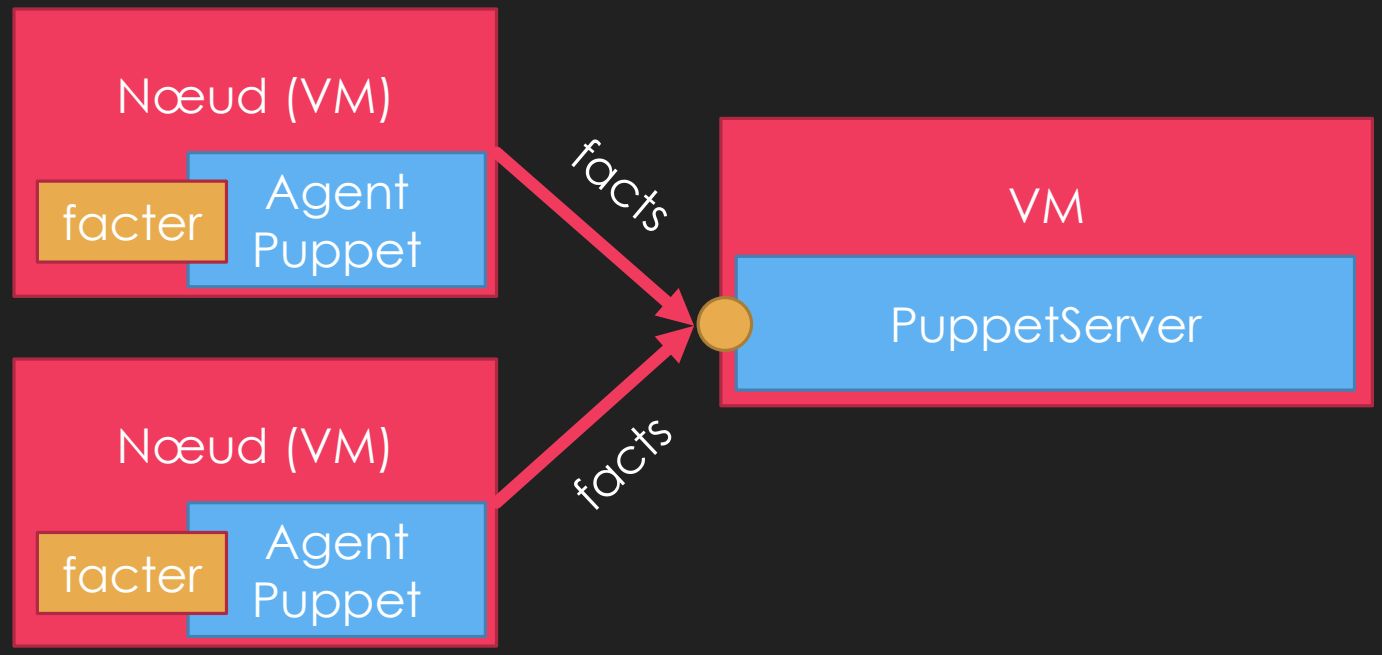

```
ensure_packages ("git")
package { "apt-listchanges": ensure => absent }
package { 'bundler': provider => 'gem'}

# Modules officiels

# puppet module install puppetlabs-apache
class { 'apache':default_vhost => false,}
apache::listen { "80": }
apache::mod { ['auth_cas']: }

# puppet module install puppetlabs-mysql
class { '::mysql::server':
    root_password    => 'x',
}
```

Chronologie d'évolutions: extension

- Problématique: extension à tout le parc
  - Différents hébergements
  - OS ou versions d'OS différents (Debian 6 -> 10, CentOS/RedHat)
  - Eviter de dupliquer le code et backporter..
- Solution: utilisation de l'environnement et des structures offertes par le langage

Chronologie d'évolutions: extension

- Utilisable de variables d'environnement
  - Facter
    - Informations OS
    - Ressources
    - Montages
    - Réseau
    - Etc
    - … et les votres !
      - « Facts » personnalisés via scripts »
      - Ex: needreboot, statut m-à-j



Chronologie d'évolutions: extension

Université Fédérale
Toulouse Midi-Pyrénées

```
$ facter --puppet
os => {
  architecture => "amd64",
  distro => {
    codename => "buster",
    description => "Debian GNU/Linux 10 (buster)",
    id => "Debian",
    release => {
      full => "10",
      major => "10"
    }
  },
  family => "Debian",
  hardware => "x86_64",
  name => "Debian",
  release => {
    full => "10.1",
    major => "10",
    minor => "1"
  },
  selinux => {
    enabled => false
  }
}
```

- Adaptation à Debian 10 de la classe générique Debian

```
ensure_packages ("apt-transport-https")

if versioncmp($lsbmajdistrelease,'10') >= 0 {
    ensure_packages ("monitoring-plugins-standard")
} else {
    ensure_packages ("nagios-plugins-standard")
}

apt::source { 'debian_main_univ':
    location => 'http://packages.univ-toulouse.fr/debian/',
    release  => $facs['os']['distro']['codename'],
    repos    => 'main contrib non-free'
}
```

Chronologie d'évolutions: extension

- Les templates: générez vos fichiers de configuration ! Prenant en compte variables environnement & variables (syntaxe Ruby ERB)

/modules/ut_debian_base/manifests/init.pp

```
class ut_debian_base (
    Boolean       $autoRestartSvcEnableCheck = true,
    Array[String] $monitor_additionnal_authorized_hosts
) {
    file {"/etc/nagios/nrpe.cfg":
        contents => template("ut_debian_base/etc/nagios/nrpe.cfg")
    }
}
```

/modules/ut_debian_base/templates/nrpe.cfg

```
allowed_hosts=127.0.0.1,<% @monitor_additionnal_authorized_hosts.each do | h | %>,<%= h %><% end %>
<% if scope.function_versioncmp([@lsbmajdistrelease,'9']) >= 0 %>
  command[check_restart]=sudo /usr/sbin/needrestart -p -k <% if @autoRestartSvcEnableCheck %>-l<% end %>
<% else %>
  command[check_checkrestart]=/opt/comue/nagios/check_checkrestart
<% end %>
```

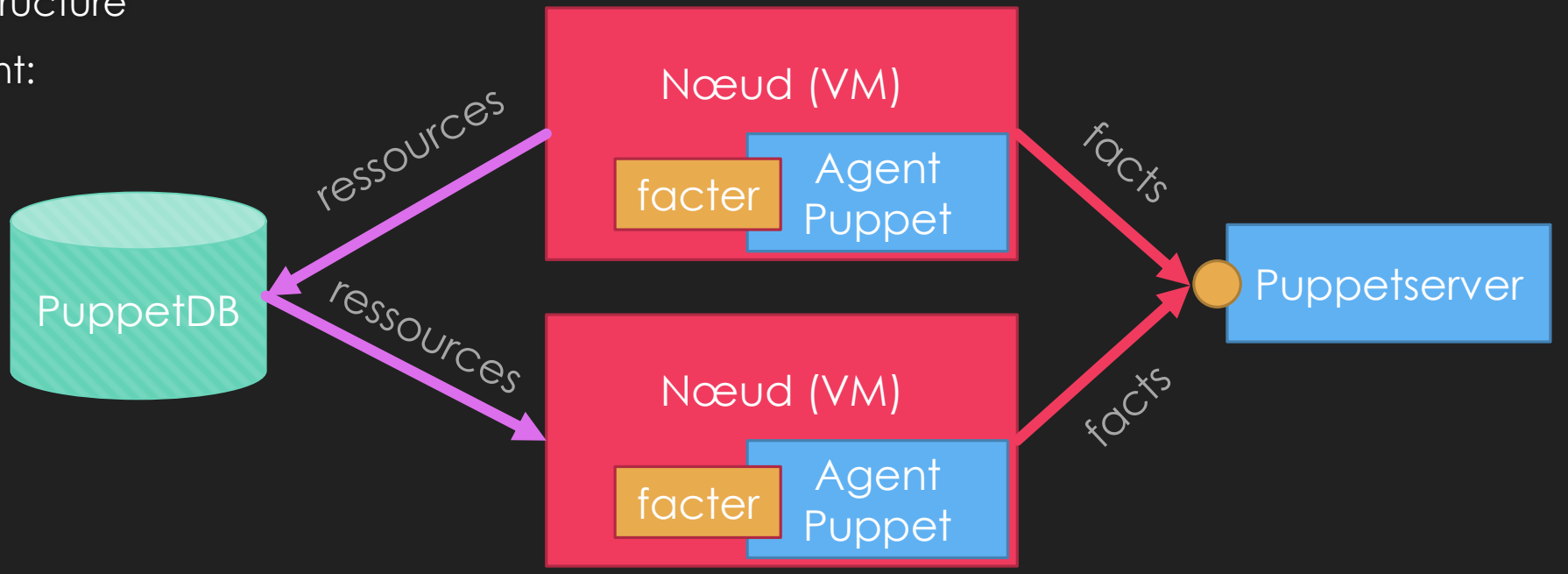
Chronologie d'évolutions: extension

- On arrive à gestion intégrale du système
  - Mises à jour automatiques, tâches planifiées
  - Accès SSH (gestion des clés)
  - Sécu / optimisation sysctl
    - En fonction des ressources de la machine
  - Motd
  - ZSH
  - Logs (syslog/rotate)
  - Désactivation IPv6 ..

```
cron { 'base_ldap_save':
     command => '(/usr/sbin/slapcat > /backup/dc=unr,dc=fr-$(date
+\'\%Y-\%m-\%d\').ldif); gzip /backup/dc=unr,dc=fr-$(date +\'\%Y-\%m-
\%d\').ldif',
     user    => 'root',
     hour    => 0,
     minute  => 10,
}
```

Chronologie d'évolutions: avancé

- Nouveau niveau d'automatisation: vision multi-machines
  - Évolution de l'infrastructure
  - Nouveau composant:
    - PuppetDB



(schéma simplifié, en réalité, les nœuds interagissent avec PuppetDB via PuppetServer)

Chronologie d'évolutions: avancé

- Nouveau niveau d'automatisation: vision multi-machines
  - On peut désormais écrire:

```
# Export vers PuppetDB
@@sshkey { $::hostname:
    type => ecdsa,
    key  => $::sshecdsakey,
    tag  => $machineGroup
}

# Import(Collecte) / Exécution des ressources
Sshkey <<| tag == $machineGroup |>>
```

Ressource exportée

Collecte

  - Ce code a pour effet de permettre à toutes les machines d'un même groupe de se connecter entre elles via SSH

Chronologie d'évolutions: avancé

- Nouveau niveau d'automatisation: vision multi-machines
  - Focus sur deux « design pattern »

Monitoring	Reverse-proxying + Web

Plan

- Introduction: du besoin à Puppet
- Architecture de base
- Chronologie d'évolutions
- **Focus sur 2 design patterns**
- Bilan d'usage

Focus sur la supervision

- Automatisation de la supervision
  - Objectifs une machine Puppetisée est monitorée
    - Sortir d'une gestion manuelle du monitoring

```
class ut_debian_base() {
    @@nagios_host { $::fqdn:
        ensure  => present,
        alias   => $::hostname,
        address => $::ipaddress,
        use     => 'generic-host',
      }
    @@nagios_service { "check_disk_${::hostname}":
      use                 => 'generic-service',
      host_name           => $::fqdn,
      check_command       => 'check_nrpe!check_disk',
      target              => '/etc/nagios3/conf.d/nagios_service.cfg',
      notify              => Service[$nagios::params::nagios_service],
    }
}
```

```
node 'orwell.univ-toulouse.fr' {
    # Machine de supervision

    # Récupération des machines à monitorer
    Nagios_host    <<| |>>

    # Récupération des services à monotorer
    Nagios_service <<| |>>

}
```

Focus sur la supervision

- Automatisation de la supervision
  - Création de fonctions-raccourcis

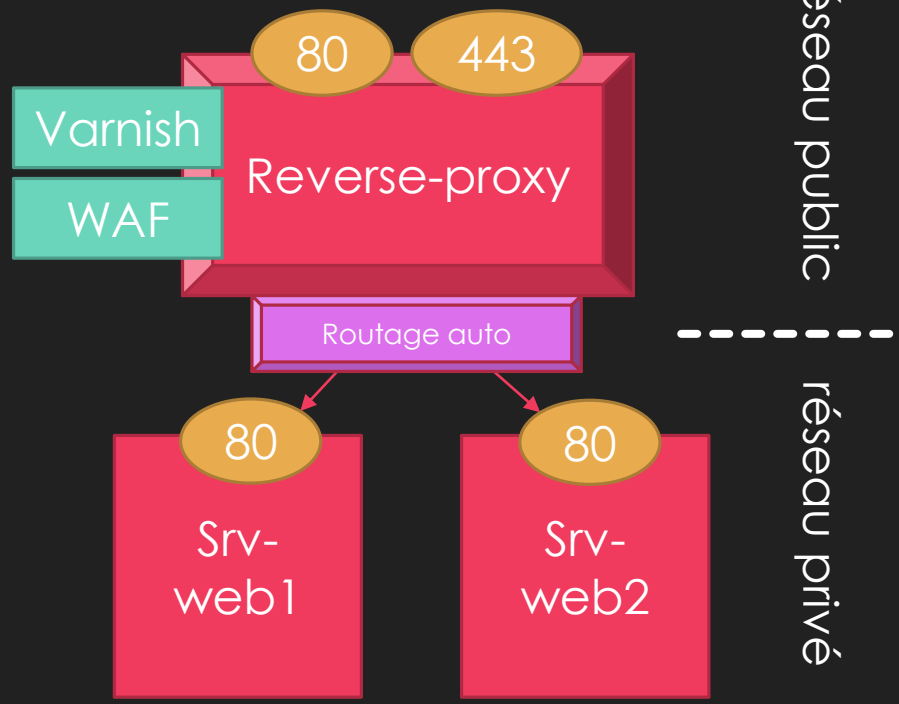
```
# Sites Web
class ut_website (){
    ut_shinken::check::disk {'/web': }
    ut_shinken::check::service {'apache2': }
    ut_shinken::check::service {'php${packageVersion}':
}
}

define ut_website::instance(
    String $sitename, $with_ssl = false,
) {
    ut_shinken::check::http {"$sitename": }
    if $with_ssl {
        ut_shinken::check::certificate {"$sitename": }
    }
}
```

Focus sur la supervision

- Automatisation de la supervision
  - Bilan: un case tête de moins
    - Les hôtes et services sont automatiquement déclarés sur le (bon) serveur de monitoring
    - Les déclarations de configuration et de supervision sont situées au même endroit
      - Et tiennent parfois sur une seule ligne !
    - 100% monitoring géré via Puppet aujourd'hui
    - Adaptations pour serveurs de test

Focus sur l'environnement Web

- Automatisation des environnements Web
  - Problématiques d'un environnement complexe:
    - Cycle de vie, maintenance, supervision des environnements Web
    - Simplification la gestion des flux entrants Web (IP / accès public)
      - Certificats et configuration TLS
      - WAF & Caching
      - Déclarations DNS, migration des services
  - => Installation d'un reverse proxy automatisé avec Puppet



Focus sur l'environnement Web

- Automatisation des environnements Web
  - Déclaration d'un backend via Puppet
  - Il reste à:
    - Générer un certificat
    - DNS
    - Sauf domaine de test ;)

```
node 'iemanja.univ-toulouse.fr' {
    @@ut_loadbalancer::backend {$::fqdn:
        tag            => "ha_COMUE-PROD",
        with_varnish   => false,
        with_waf       => true,
        restricted_ips => [],
        sites          => [{
                title   => "www.univ-toulouse.fr",
                to_ssl  => true
                aliases => ['en.univ-toulouse.fr','univ-
    toulouse.fr']
        }]
    }
}
```

Focus sur l'environnement Web

- Automatisation des environnements Web
  - Une seule déclaration d'un site PHP-MySQL
    - Crée un vhost Apache, un service PHP fpm
    - Crée la base de donnée SQL
    - Déclare sur le reverse proxy
    - Surpervise
    - Gère les ACLs

```
node 'iemanja.univ-toulouse.fr' {
    ut_debian_website::php_instance {'siteinst':
        sitename  => "www.univ-toulouse.fr",
        aliases   => ['en.univ-toulouse.fr','univ-toulouse.fr'],
        memory_limit                => '320M',
        declare_to_haproxy          => true,
        plain_redirect_to_secure    => true,
        users                       => ['chanouha','x','y'],
        database_user               => "x",
        database_name               => "x",
        database_password           => "x",
    }
}
```

Plan

- Introduction: du besoin à Puppet
- Architecture de base
- Chronologie d'évolutions
- Focus sur 2 design patterns
- **Bilan d'usage et conclusion**

Bilan d'usage: déclaration-type e

- Une déclaration complète de machine Debian ressemble à:

- Et Windows ?
  - Peu de classes spécifiques à Windows
  - Mais les fonctions de bases fonctionnent (Fichier / Service)
  - Utilisation pour la surpervision essentiellement

```
node 'iemanja.univ-toulouse.fr' {

    # Machine de production web Debian Stretch, ip privée
        $contacts    = [Ut_comue::User['x']]
        $responsable = Ut_comue::User['chanouha']
        $description = "Nextcloud"
        $entite      = "INFRA/OUTILS"

        class { 'ut_debian_base':
            with_standart_root_pw => true,
            autoReboot            => true,
        }
        class { 'ut_debian_autoupdate':
            update_hour     => 8,
            updates         => "SECURITY"
        }
        require ut_cloud
        require ut_mail_smarthost_exim4
}
```

Bilan d'usage

Université Fédérale
Toulouse Midi-Pyrénées

- Etat de l'installation
  - Intégralité du parc de 130 machines
    - 85% Debian 6 => 10, 10% CentOS/RedHat 6 => 8, < 5% Windows
  - 1 serveur Puppet nécessaire: 4 cPU, 4,5 Go RAM, 16go disque suffit
  - Des modifications tous les jours
- Qualité de service: simplicité et stabilité
  - Au moindre souci, le catalogue ne s'exécute pas, évitant de potentiels problèmes
  - Aucun incident lié à l'agent Puppet ou la mise à jour de dépôt officiels (et mêmes contribs !) qui sont globalement de très bonne facture
  - Incidents souvent liés à une mauvaise syntaxe, ou paramètre non adapté à telle version d'OS/App (coucou Apache)

Bilan d'usage

Université Fédérale
Toulouse Midi-Pyrénées

- Répond très bien aux besoins COMUE qui a peu besoins structurants
  - Majorité demandes = Environnements Web, Mise à dispositions de VM pour applicatifs spécifiques

- Petits bémols:
  - Un accès Puppet = un accès root sur toutes les machines (pas de cloisonnement)
  - Outils fait pour l'infrastructure, moins adapté pour travail inter-équipes (écriture de code)
  - Consommation mémoire de l'agent (lors de l'application du catalogue)
  - Visibilité/Reporting du parc dans la version OpenSource
    - Compensé par développements (export PuppetDB) + Monitoring avancé

Jusqu'à où peut-on aller ?

- Ce que ne j'ai pas mis en place
  - Les fonctionnalités entreprises (Puppet Open Source)
    - Node classifier (remplace les manifests de nœuds)
    - Orchestration
    - Reporting
  - Gestion plus industrialisée des déclaration de nodes (hiera)

- Ne remplace pas l'intégration continue, mais permet de mettre en place le processus
  - En complément de Gitlab,
- Flexible, mais non adapté à besoins temps réels et évolutifs
  - Des architectures conteneur seraient plus adaptées

Fin de la présentation.

Des questions ?