



CURSO 525

INFRAESTRUTURA ÁGIL COM PRÁTICAS DEVOPS



[illegible]

- ## Anotações



O Puppet é uma ferramenta de gerenciamento de configurações de sistemas Unix-like e Windows. Através de uma linguagem declarativa, é possível descrever todo o ambiente, desde quais pacotes deverão ser instalados, quais versões que deverão se encontrar, qual a configuração, entre outros.

Puppet foi desenvolvido como uma solução para o problema de manter grandes parques de máquinas, com exatamente a mesma configuração. Antes de sua criação, era um trabalho muito complexo manter a configuração e até mesmo análise de erros, visto que cada máquina em seu ambiente estava com uma configuração diferente, ocasionando na solução, ser na maioria das vezes 'Formatar a Máquina'.

Atualmente, a ferramenta é muito utilizada no mercado, apesar de possuir concorrentes como o Ansible, Chef, Saltstack, entre outros. Em algumas ambientes é até possível encontrar o Puppet com mais uma ferramenta de gerenciamento de configurações, muito dos casos sem o Ansible. Obviamente nesse caso, o Ansible é mais utilizado como uma forma de deploy de aplicações, enquanto o Puppet serve para manter.

Uma dúvida que surge muito, é qual a diferença do Puppet para o Ansible, ou até mesmo se é realmente necessário aprender os dois. A maior diferença, além do fato do Ansible ser agentless, é que a ferramenta não é totalmente idempotente, ou seja: a maioria dos comandos do Ansible irão executar N vezes, mesmo que na primeira vez já tenha sido alterado.

Linguagem Declarativa

Possui uma linguagem para desenvolvimento de módulos muito semelhante a C, que permite descrever todo o ambiente de forma fácil.

Comunicação via SSL

Toda a comunicação entre os agents e o master é feita através de SSL, garantido a segurança dos dados que são trafegados entre as máquinas.

[illegible]

Sua infraestrutura agora estará com a mesma configuração, não importa o que aconteça. Caso uma mudança seja realizada na máquina, o agent replicará a configuração novamente.

Com o puppet, conseguimos automatizar a aplicação de configuração em diversas máquinas ao mesmo tempo.

Diferente do Ansible, o Puppet possui todos os módulos funcionando para qualquer distribuição e/ou sistema operacional.

This image shows a blank sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

Puppet Autônomo x Puppet Server

Puppet Autônomo

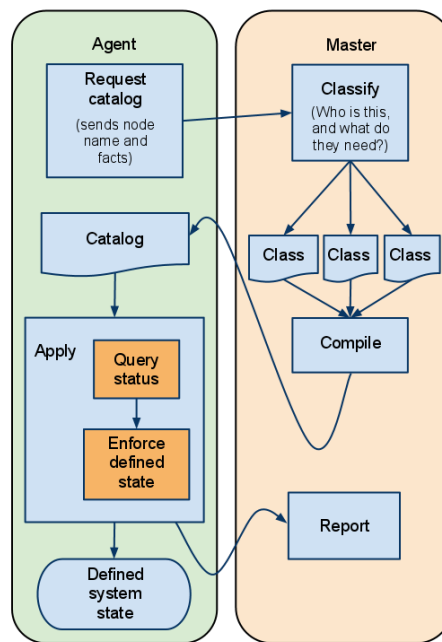
É o modelo onde instalamos os módulos do Puppet em cada máquina, e então deixamos que o serviço do agent prossiga executando os passos dentro do arquivo. Funcionaria de certa forma, que nem o Ansible, porém nesse caso o Autônomo só executa localmente.



Puppet Server

Possui um serviço central, cuja responsabilidade é validar o certificado de cada agent, e então repassar os módulos requisitados. Nesse modelo, você precisa garantir que o módulo tenha sido enviado para o servidor, somente garanta que o agent esteja sendo executado.

Diferente do Ansible, a palavra módulo se refere a um conjunto de recursos com o objetivo de configurar algo na máquina hospedeira. Seria o equivalente as roles que vimos anteriormente durante a aula de Ansible.



Um ambiente em que se utiliza o Puppet Master é bem simples: o agent irá requisitar os módulos que deverão ser executados na máquina hospedeira, e informará em qual ambiente ele se encontra, como também seu hostname e seu certificado.

Caso os dados sejam considerados válidos, o master irá conferir se dentro de seu catálogo existem módulos para esse agent. Caso existam, eles serão compilados e então enviados para o agent, que passará a aplicar cada uma dessas configurações.

Ao término, será definido um status para o servidor e então enviado um relatório da execução para o servidor master.

Recursos é como são chamados as ações que você pode executar numa máquina de destino, sendo desde instalar um pacote, alterar um arquivo, executar um comando, entre outros. Recursos abstraem a necessidade de saber o sistema operacional da máquina de destino.

Package	Service	User
✓ gems;	✓ systemd;	✓ useradd;
✓ yum;	✓ service;	✓ ldapadd;
✓ apt;	✓ launchd;	✓ adduser;
✓ deb;	✓ upstart;	✓ Netinfo.
✓ rpm;		

Assim como o Ansible, o Puppet também possui milhares de recursos disponíveis para o uso, e uma documentação para descrever o que fazem e os parâmetros de cada um. Alguns dos resources mais comuns que temos são:

service: Responsável por gerenciar os serviços do sistema;

user: Responsável por gerenciar as contas de usuários do sistema;

mount: Responsável por gerenciar os sistemas de arquivos e o arquivo fstab;

group: Responsável por gerenciar os grupos de usuários do sistema;

exec: Executa comandos no servidor remoto, contanto que o comando em questão não seja uma shell.

package: Instalação e remoção de pacotes do servidor, não importando a distribuição;

cron: Instalação e gerenciamento de tarefas agendadas;

file: Gerencia todos os tipos de arquivos do sistema, desde texto até diretórios.

- ## Anotações

This image shows a blank sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.



Usando o Puppet em modo autônomo e escrevendo Manifestos

Anotações

[illegible]

- ## Anotações

Para testar o Puppet acesse a máquina compliance.

```
# cd infraagil
```

```
# vagrant ssh compliance
```

```
# sudo su -
```

```
# puppet --version
```

compliance.4labs.example

Anotações

This image shows a blank sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

Modo Autônomo



Primeiramente, iremos aprender como é o funcionamento do puppet em modo autônomo, para isto devemos entender os recursos disponíveis como: **service**, **user**, **group**, **exec**, **package**, **cron** e **file**.



service: Responsável por gerenciar os serviços do sistema;

user: Responsável por gerenciar as contas de usuários do sistema;

mount: Responsável por gerenciar os sistemas de arquivos e o arquivo fstab;

group: Responsável por gerenciar os grupos de usuários do sistema;

exec: Executa comandos no servidor remoto, contanto que o comando em questão não seja uma shell.

package: Instalação e remoção de pacotes do servidor, não importando a distribuição;

cron: Instalação e gerenciamento de tarefas agendadas;

file: Gerencia todos os tipos de arquivos do sistema, desde texto até diretórios.

Vamos utilizar o recurso de gerenciamento de usuários:

```
$ puppet resource user
$ puppet resource user root
$ puppet resource user linus
$ puppet resource user linus ensure=present
$ puppet resource user linus
$ puppet resource user linus ensure=absent
$ puppet resource user linus
$ puppet resource user linus ensure=present home="/srv/linus"
managehome=true
```

compliance.4labs.example

resource user – Lista todos os usuários da máquina;

resource user <user> – Lista o usuário <user>;

resource user <user> ensure=present – Garante que o usuário <user> seja criado, caso não exista;

resource user <user> ensure=absent – Garante que o usuário <user> seja removido, caso exista.

resource user <user> ensure=present home="/srv/<user>"

managehome=true – Garante que o usuário <user> exista e que sua pasta home seja /srv/<user>.

Vamos utilizar o recurso de gerenciamento de serviços:

```
$ puppet resource service
$ puppet resource service cron
$ puppet resource service ntp
$ puppet resource service cron ensure=stopped
$ ps -aux | grep cron
$ puppet resource service cron ensure=running
$ ps -aux | grep cron
```

compliance.4labs.example

puppet resource service – Lista todos os serviços da máquina;

puppet resource service <service> - Lista o serviço <service>;

puppet resource service <service> ensure=stopped – Garante que o serviço esteja parado;

puppet resource service <service> ensure=running – Garante que o serviço esteja em execução.

Vamos utilizar o recurso de gerenciamento de pacotes:

```
$ puppet resource package
$ puppet resource package cron
$ puppet resource package ntpdate
$ dpkg -l | grep ntpdate
$ puppet resource package ntpdate ensure=present
$ dpkg -l | grep ntpdate
$ puppet resource package ntpdate ensure=absent
$ dpkg -l | grep ntpdate
```

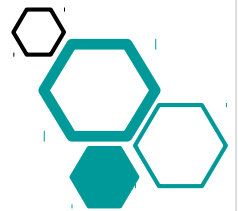
compliance.4labs.example

puppet resource package – Lista todos os pacotes da máquina;

puppet resource package <package> - Lista o pacote <package>;

puppet resource package <package> ensure=present – Garante que o pacote esteja presente;

puppet resource package <package> ensure=absent – Garante que o pacote esteja ausente.



Anotações

This image shows a blank sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

1

Verifique se o factor está instalado:

factor --version

2

Execute o comando para listar todas as informações do sistema:

factor

3

Execute o comando para consultar o hostname da máquina:

factor hostname

4

Execute o comando para consultar as interfaces da máquina:

factor interfaces

5

Execute o comando para consultar o endereço IP de uma determinada interface:

factor ipaddress_<interface>

compliance.4labs.example

Os parâmetros possíveis para filtro do factor podem ser encontrados em:
https://puppet.com/docs/facter/3.11/core_facts.html

Manifests



Quando precisamos de ações mais complexas do que executar um único comando, utilizamos Manifests.

Manifests são basicamente uma coleção de declarações de recursos, utilizando a extensão **.pp**.



Por padrão, a pasta de manifests localizada em `/etc/puppetlabs/code/environments/production/manifests` é utilizada pelo puppet master (server), veremos sobre este assunto na próxima aula.

Vamos criar nosso primeiro manifest para adicionar um usuário.

1

Crie os diretórios `/opt/puppet/`:
`# mkdir -p /opt/puppet/`

2

Gere o arquivo **users.pp** utilizando o modelo da pasta Aula 7.2:
`# sudo cp /vagrant/4525/arquivos/Aula\ 7.2/users.pp /opt/puppet/`
`# cat /opt/puppet/users.pp`

```

user { 'linus':
  ensure => 'present',
  password => '$1$9MCsbH8/$1sUqPu61bCSpaWIkAJIr51',
  home    => '/srv/linus',
}
(...)

```

compliance.4labs.example

Para criar a senha criptografada, podemos utilizar o comando **openssl passwd -1**

==== users.pp =====

```

user { 'linus':
  ensure => 'present',
  password => '$1$9MCsbH8/$1sUqPu61bCSpaWIkAJIr51',
  home    => '/srv/linus',
}
user { 'analista':
  ensure => 'present',
  password => '$1$9MCsbH8/$1sUqPu61bCSpaWIkAJIr51',
  home    => '/srv/analista',
}

```

compliance.4labs.example

```
3 # exit
# su analista
4linux
# exit
```

Anotações

Manifests



Podemos utilizar relacionamentos e ordenações para dizer que um recurso deve ser executado antes ou após outro recurso.

Para isto, podemos utilizar os parâmetros **require**, **before**, **notify** e **subscribe**, ou então **utilizar setas de encadeamento**.



before: Aplica um recurso antes do recurso alvo;

require: Aplica um recurso depois do recurso alvo;

notify: Aplica um recurso antes do recurso alvo, o alvo é atualizado se o recurso notify for modificado;

subscribe: Aplica um recurso depois do recurso alvo, o recurso subscribe é atualizado se o recurso alvo for modificado.

Para exemplificar o uso dos parâmetros de relacionamento e ordenação, vamos instalar um webserver e garantir que o serviço esteja habilitado e executado.

1

Troque para o usuário root:

```
# sudo su -
```

2

Gere o arquivo **nginx.pp** utilizando o modelo da pasta Aula 7.2:

```
# sudo cp /vagrant/4525/arquivos/Aula\ 7.2/nginx.pp /opt/puppet/  
# cat /opt/puppet/nginx.pp  
package {'nginx':  
  ensure => installed,  
  before => Service['nginx']  
}  
(...)
```

compliance.4labs.example

```
=== nginx.pp ===
```

```
package {'nginx':  
  ensure => installed,  
  before => Service['nginx']  
}
```

```
service {'nginx':  
  ensure => running,  
  enable => true,  
  require => Package['nginx']  
}
```

```
Execute o manifest nginx.pp:
# puppet apply /opt/puppet/nginx.pp
```

Verifique se o nginx está habilitado e se é funcional:

```
# curl compliance.4labs.example
```

Anotações

Agora, vamos criar um novo manifest utilizando as setas de encadeamento.

1

```
Gere o arquivo ntp.pp utilizando o modelo da pasta Aula 7.2:
# sudo cp /vagrant/4525/arquivos/Aula\ 7.2/ntp.pp /opt/puppet/
# cat /opt/puppet/ntp.pp
package {'ntp':
  ensure => installed,
}
-> file { '/etc/ntp.conf':
  ensure => file,
  mode   => '0600',
  source => '/vagrant/files/ntp.conf'
}
(...)
```

compliance.4labs.example

A seta de ordenação é um hífen e sinal de maior que `->` . Aplica o recurso da esquerda antes do recurso da direita.

A seta de notify é um til e um sinal de maior que `~>` . Aplica o recurso da esquerda antes, se o recurso modificar algo, o recurso da direita é aplicado.

Package['ntp'] -> File['/etc/ntp.conf'] ~> Service['ntp']

→ Execute o Bloco Package, então execute o bloco File e após sua modificação, execute o bloco Service.

=== ntp.pp ===

```
package {'ntp':
  ensure => installed,
}
-> file { '/etc/ntp.conf':
  ensure => file,
  mode   => '0600',
  source => '/vagrant/files/ntp.conf'
}
~> service {'ntp':
  ensure => running,
  enable => true,
}
```

Configuramos para que seja feita a instalação do serviço do NTP e que somente após a instalação, fosse copiado o arquivo `ntp.conf` e em seguida, habilitado e iniciado o serviço.

- 1 Execute o manifest ntp.pp:

```
# puppet apply /opt/puppet/ntp.pp
```
- 2 Verifique se o serviço do ntp foi habilitado e está sendo executado:

```
# puppet resource service ntp
```
- 3 Verifique se o arquivo ntp.conf foi copiado:

```
# head /etc/ntp.conf
```

compliance.4labs.example

Anotações

[illegible]

- 1 Testando a instalação do Puppet
- 2 Modo Autônomo
- 3 Facter
- 4 Manifests

Anotações

[illegible]

This image shows a blank sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

- ## Anotações



É importante lembrar que a arquitetura do puppet é **Master to Agent**, sendo assim quem efetua a solicitação é o Puppet Agent.

This image shows a blank sheet of white paper with horizontal ruling lines. The lines are evenly spaced and extend across the width of the page. There are no margins, text, or other markings on the paper.

Para garantir o endereço do puppetserver, precisamos adicionar uma entrada no arquivo puppet.conf.

```
1  # vagrant ssh compliance
   # sudo su -
```

```

2  [main]
    server = compliance.4labs.example
    dns_alt_names = compliance.4labs.example

```

compliance.4labs.example

Anotações

[illegible]

1

2

3

5

Agora, precisamos habilitar o puppet agent que será responsável por aplicar os manifestos e se comunicar com o puppetserver.

```
1 # puppet agent -t
```

```
2 # puppet resource service puppet ensure=running enable=true
```

3 Verifique os status do serviço do puppet agent:
systemctl status puppet

compliance.4labs.example

Anotações

This image shows a blank sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

[illegible]

Agora, vamos conectar a máquina automation ao puppet server para que seja gerado um certificado.

Em outro terminal ou janela, conecte via ssh na máquina automation:

```
1 # cd infraagil
# vagrant ssh automation
```

Execute a playbook do puppet agent apenas na máquina automation:

```
2 # sudo ansible-playbook -l automation
/etc/ansible/playbooks/puppet-agent.yml
```

Após a execução, será exibido um erro na task “Executando o Puppet Agent nos Servidores”, isto deve-se ao fato de ser necessário efetuar a assinatura manual do certificado.

automation.4labs.example

Anotações

[illegible]

Na máquina automation, podemos verificar a configuração executando novamente o `playbook` ou executando o comando `puppet agent -t`.

```
1 # sudo ansible-playbook -l automation
   /etc/ansible/playbooks/puppet-agent.yml
```

```
2 # bash
# puppet agent -t --server compliance.4labs.example
```

automation.4labs.example

Anotações

This image shows a blank sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

Na máquina **compliance** é possível revogar e excluir o certificado:

compliance.4labs.example

Caso o nó tenha que se conectar novamente ao puppet master, é necessário remover o conteúdo da pasta ssl.

Na máquina **automation**, mova o conteúdo da pasta ssl:

automation.4labs.example

Anotações

É possível criar um arquivo de autosign com o nome das máquinas que terão seus certificados assinados automaticamente.

1 Alterne para a máquina **compliance** e crie o arquivo `autosign.conf`:

```
# vim /etc/puppetlabs/puppet/autosign.conf
automation.4labs.example
```

compliance.4labs.example

Anotações

[illegible]

automation.4labs.example

```
2 # puppet agent -t --server compliance.4labs.example
```

Anotações

Altere para a máquina **compliance** é exclua novamente o certificado:

```
# puppetserver ca clean --certname automation.4labs.example
```

Agora que servidor possui o arquivo "autosign.conf" o nó precisa se conectar novamente ao puppet master.

automation.4labs.example

Anotações

Retornando agora na máquina compliance, poderemos verificar que o certificado já foi assinado.

1 Liste os certificados:
`# puppetserver ca list --all`

Vamos editar nosso arquivo autosign para que ele assine automaticamente todas as máquinas do domínio 4labs.example.

2 Edite o arquivo de autosign:
`# vim /etc/puppetlabs/puppet/autosign.conf`
`*.4labs.example`

compliance.4labs.example

Para o conteúdo do arquivo autosign, podemos utilizar o nome de cada servidor ou então, utilizar uma regex como ***.4labs.example** para dizer que todas as máquinas de um determinado domínio terão seus certificados autoassinados.

ex:

/etc/puppetlabs/puppet/autosign.conf

automation.4labs.example

log.4labs.example

container.4labs.example

scm.4labs.example

Adicione a entrada **puppet** ao arquivo hosts do puppetserver para que o mesmo possa se identificar como um servidor puppet

- 1 Altere o arquivo /etc/hosts

```
# sudo vim /etc/hosts  
127.0.0.1 localhost puppet
```
- 2 Reinicie o serviço do Puppet Server

```
# sudo systemctl restart puppetserver
```

compliance.4labs.example

A entrada puppet ao arquivo /etc/hosts do servidor apontando para o ip local faz com que o servidor se reconheça no momento em que o puppetserver executar algum manifests ou efetuar a chamada **puppet:<metodos>**

- ## Anotações



This image shows a blank sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

- 

[illegible]



Através dos módulos, podemos agrupar as configurações em um único local, sendo possível reaproveitar e otimizar nosso código.

This image shows a blank sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

Acesse a máquina compliance e troque para o usuário root::

```
2 # puppet resource package pdk ensure=present
```

compliance.4labs.example

[illegible]

Crie o módulo puppet-inspec através do seguinte comando:

1

```
Metadata will be generated based on this information, continue?
(Y/n) Y
```

Anotações

Vamos criar o arquivo `init.pp` na pasta `manifests` do módulo `inspec`, adicionar as instruções de download e instalação.

Gere o arquivo **init.pp** utilizando o modelo **inspec_init.pp** da pasta Aula 7.4:

```
# cp /vagrant/4525/arquivos/Aula\ 7.4/manifests/inspec_init.pp
inspec/manifests/init.pp
# cat inspec/manifests/init.pp
class inspec {
    file {'/opt/inspec.deb':
        path      => '/opt/inspec.deb',
        source     =>
        'https://packages.chef.io/files/stable/inspec/4.12.0/ubuntu/18.04/
inspec_4.12.0-1_amd64.deb',
        before    => Package['inspec']
    }
    (...)
}
```

compliance.4labs.example

O arquivo `init.pp` é o primeiro arquivo a ser procurado em um módulo do puppet.

=== init.pp ===

```
class inspec {
```

```
    file {'/opt/inspec.deb':
```

```
        path      => '/opt/inspec.deb',
```

```
        source     => 'https://packages.chef.io/files/stable/inspec/4.12.0/ubuntu/18.04/
```

```
inspec_4.12.0-1_amd64.deb',
```

```
        before    => Package['inspec']
```

```
    }
```

```
    package {'inspec':
```

```
        ensure    => installed,
```

```
        provider  => dpkg,
```

```
        source    => '/opt/inspec.deb',
```

```
        require   => File['/opt/inspec.deb']
```

```
    }
```

```
}
```

Para que um módulo seja aplicado, devemos criar o arquivo site.pp com as definições dos nós.

Crie o arquivo site.pp com o seguinte conteúdo:

```
# vim /etc/puppetlabs/code/environments/production/manifests/site.pp

1 node 'compliance' {
    include inspec
}

node 'default' {
}
```

compliance.4labs.example

O arquivo site.pp, é onde definimos quais módulos devem ser aplicados em cada nó que possua o puppet-agent configurado e sendo executado, para garantir a configuração da infraestrutura como código.

O arquivo segue o padrão:

```
node '<NOME DA MAQUINA>' {
    include <MODULO>
    include <MODULO>
}
```

Podemos declarar também um nó com o nome **default**, isto faz com que as configurações do bloco default sejam aplicadas a todos os servidores que se conectarem ao Puppet Master e não possuírem definições.

Obs.: O bloco default deve ser o ultimo bloco, o puppet lê o arquivo de cima para baixo, uma vez que um nó deu **match** com o nome, é feita a compilação do catálogo e aplicação das configurações.

Caso necessário, podemos utilizar o comando puppet parser validate para verificar a sintaxe do nosso código.

```
puppet parser validate /etc/puppetlabs/code/environments/production/modules/inspec/
manifests/init.pp
```

Criação de Módulos

A configuração será aplicada de tempos em tempos, conforme configurada no arquivo `/etc/puppetlabs/puppet.conf`. Por padrão, a configuração é executada a cada 30 minutos.

```
[main]
server = compliance.4labs.example
runinterval = 30m
```

compliance.4labs.example

O parâmetro `runinterval` pode ser modificado para minutos, segundos, horas, dias ou anos.

Caso não seja especificado a unidade, o puppet assume que o valor inserido é em segundos.

ex:

<code>runinterval = 10</code>	10 Segundos
<code>runinterval = 30s</code>	30 Segundos
<code>runinterval = 45m</code>	45 Minutos
<code>runinterval = 6h</code>	6 Horas
<code>runinterval = 2d</code>	2 Dias
<code>runinterval = 5y</code>	5 Anos

compliance.4labs.example

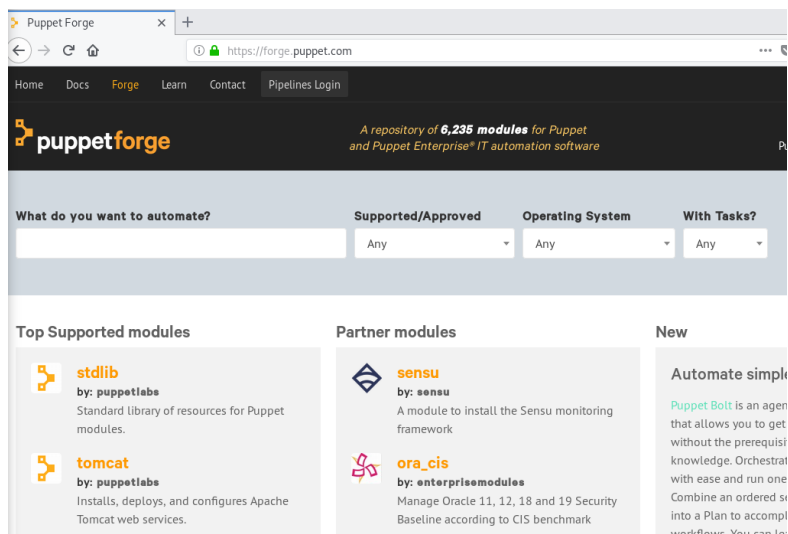
- | | |
|---|--|
| 1 | Force a aplicação da configuração:
<code># puppet agent -t</code> |
| 2 | Após a aplicação do catálogo, verifique se o inspec foi instalado corretamente:
<code># inspec --version</code> |

Anotações

This image shows a blank sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.


This image shows a blank sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

<https://forge.puppet.com>.



Anotações

[illegible]



puppet **orange**

A repository of **6,236 modules** for Puppet
and Puppet Enterprise* IT automation software

What do you want to automate?

elastic

Supported/Approved

Any

Operating System

Any


With Tr

Any

Found 49 modules matching 'elastic'

Filter by Puppet version: Puppet/Puppet Enterprise Version

Sort by: Relevancy



elastic_stack
elastic

Helpers for installing and configuring components of the Elastic Stack.


Version 7.0.0

PKK

Updated 6 days ago

Total downloads 5,588,252

Quality score 5.0



elasticsearch
elastic

Module for managing and configuring Elasticsearch nodes

Version 6.3.4

APPROVED


Updated 11 days ago

Total downloads 1,429,662

Quality score 4.7

Anotações

This image shows a blank sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.



elastic/elastic_stack

by: Elastic

Helpers for installing and configuring components of the Elastic Stack.

[Project URL](#) [RSS Feed](#)

Latest version is compatible with:

- Puppet Enterprise 2019.1.x, 2019.0.x, 2018.1.x, 2017.3.x, 2017.2.x, 2016.4.x
- Puppet >= 4.10.0 < 7.0.0
- RedHat, Ubuntu, Debian, SLES, OpenSuSE

Start using this module:

new! Bolt

r10k or Code Manager

Manual installation

Manually install this module with Puppet module tool:

```
puppet module install elastic-elastic_stack --version 7.0
```

[Report issues](#)

This image shows a blank sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

Pesquise pelo módulo `elastic_stack`:

```
# puppet module search elastic_stack
```

2

Instale o módulo `elastic-elastic_stack`:

```
# puppet module install elastic-elastic stack --version 7.0
```

Anotações

- 

[illegible]



This image shows a blank sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

Escrevendo Módulos no Puppet

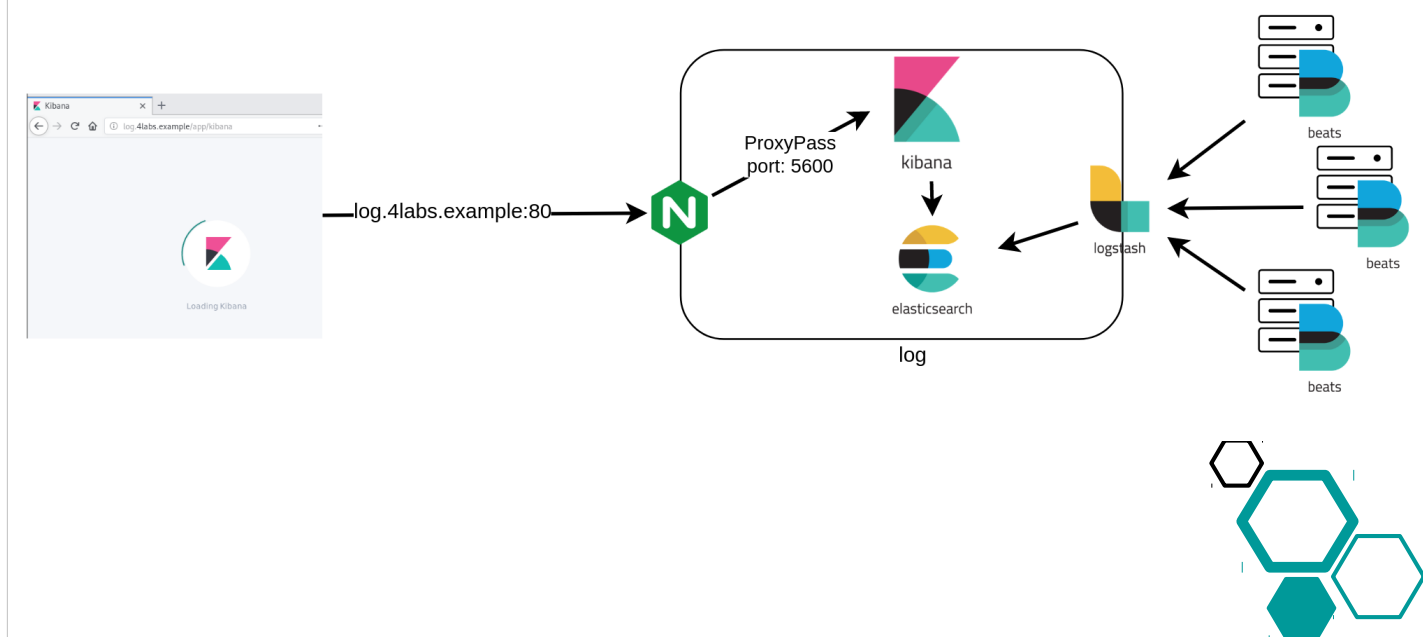
Anotações

This image shows a blank sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.



Anotações

This image shows a blank sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.



Anotações

This image shows a blank sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

Escrevendo Módulos – Java 8

Vamos criar um manifest para a instalação do Java8, que é uma das dependências necessárias para a instalação do ELK. Criaremos todos os manifests dentro da pasta do módulo do ELK.

1

Na máquina compliance acesse o diretório dos módulos do puppet:

```
# cd /etc/puppetlabs/code/environments/production/modules
```

2

Crie o manifest para a instalação do Java 8 utilizando o modelo da pasta Aula 7.4:

```
# cp /vagrant/4525/arquivos/Aula\ 7.4/manifests/java8.pp
elastic_stack/manifests/
# cat elastic_stack/manifests/java8.pp
class elastic_stack::java8 {
  package {'openjdk-8-jre-headless':
    ensure => installed,
  }}
}
```

compliance.4labs.example

```
===== java8.pp =====
```

```
class elastic_stack::java8 {

  package {'openjdk-8-jre-headless':
    ensure => installed,
  }
}
```

Escrevendo Módulos - Elasticsearch

Agora, iremos criar o manifest e os arquivos necessários para a instalação do elasticsearch.

1

Crie um diretório **files** dentro da pasta elastic_stack:
`# mkdir elastic_stack/files`

2

Crie o arquivo elasticsearch.yml que será responsável pela parametrização da aplicação:
`# cp /vagrant/4525/arquivos/Aula\ 7.4/files/elasticsearch.yml
elastic_stack/files/
cat elastic_stack/files/elasticsearch.yml
path.data: /var/lib/elasticsearch
path.logs: /var/log/elasticsearch
network.host: localhost
http.port: 9200`

compliance.4labs.example

===== elasticsearch.yml =====

```
path.data: /var/lib/elasticsearch
path.logs: /var/log/elasticsearch
network.host: localhost
http.port: 9200
```

Crie o manifest para a instalação do elasticsearch utilizando o modelo da pasta Aula 7.4:

```
# cp /vagrant/4525/arquivos/Aula\ 7.4/manifests/elasticsearch.pp
elastic_stack/manifests/
# cat elastic_stack/manifests/elasticsearch.pp
class elastic_stack::elasticsearch {
```

```
1   package {'elasticsearch':
      ensure => installed,
      before => Service['elasticsearch']
    }
    file { 'elasticsearch.yml':
      path      => '/etc/elasticsearch/elasticsearch.yml
    (...)
```

compliance.4labs.example

===== elasticsearch.yml =====

```
class elastic_stack::elasticsearch {

  package {'elasticsearch':
    ensure => installed,
    before => Service['elasticsearch']
  }
  file { 'elasticsearch.yml':
    path      => '/etc/elasticsearch/elasticsearch.yml',
    source => 'puppet:///modules/elastic_stack/elasticsearch.yml',
    require => Package['elasticsearch'],
  }
  file_line {'JVM-elasticsearch-XMS':
    path => '/etc/elasticsearch/jvm.options',
    line => '-Xms512m',
    match => '-Xms1g',
    notify => Service['elasticsearch']
  }
  file_line {'JVM-elasticsearch-XXM':
    path => '/etc/elasticsearch/jvm.options',
    line => '-Xmx512m',
    match => '-Xmx1g',
    notify => Service['elasticsearch']
  }
  service {'elasticsearch':
    ensure => running,
    enable => true,
    require => File['elasticsearch.yml']
  }
}
```

1

Crie o arquivo kibana.yml que será responsável pela parametrização da aplicação:

```
# cp /vagrant/4525/arquivos/Aula\ 7.4/files/kibana.yml
```

```
elastic_stack/files/
```

```
# cat elastic_stack/files/kibana.yml
```

```
server.port: 5601
```

```
server.host: "localhost"
```

```
elasticsearch.hosts: ["http://localhost:9200"]
```

compliance.4labs.example

===== kibana.yml =====

```
server.port: 5601
```

```
server.host: "localhost"
```

```
elasticsearch.hosts: ["http://localhost:9200"]
```

Crie o manifest para a instalação do kibana utilizando o modelo da pasta Aula 7.4:

```
# cp /vagrant/4525/arquivos/Aula\ 7.4/manifests/kibana.pp
elastic_stack/manifests/
# cat elastic_stack/manifests/kibana.pp
class elastic_stack::kibana {
```

```
1   package {'kibana':
      ensure => installed,
      before => Service['kibana']
    }
    file { 'kibana.yml':
      path      => '/etc/kibana/kibana.yml',
      (...)
    }
```

compliance.4labs.example

===== kibana.pp =====

```
class elastic_stack::kibana {
```

```
  package {'kibana':
    ensure => installed,
    before => Service['kibana']
  }
```

```
  file { 'kibana.yml':
    path      => '/etc/kibana/kibana.yml',
    source    => 'puppet:///modules/elastic_stack/kibana.yml',
    require   => Package['kibana'],
    notify    => Service['kibana']
  }
```

```
  service {'kibana':
    ensure => running,
    enable => true,
    require => File['kibana.yml']
  }
}
```

Vamos instalar e configurar um NGINX como proxy reverso para o kibana.

Crie o arquivo para configuração do proxy reverso NGINX utilizando o modelo da pasta Aula 7.4:

```
# cp /vagrant/4525/arquivos/Aula\ 7.4/files/default elastic_stack/
files/
# cat elastic_stack/files/default
1 server {
    listen 80;
    server_name log.4labs.example;
    location / {
        proxy_pass http://localhost:5601;
    }
}
```

compliance.4labs.example

===== default =====

```
server {
    listen 80;

    server_name log.4labs.example;
    location / {
        proxy_pass http://localhost:5601;
    }
}
```

Crie o manifest para a instalação do nginx utilizando o modelo da pasta Aula 7.4:

```
# cp /vagrant/4525/arquivos/Aula\ 7.4/manifests/nginx.pp
elastic_stack/manifests/
# cat elastic_stack/manifests/nginx.pp
class elastic_stack::nginx {

  1   package {'nginx':
        ensure => installed,
        before => Service['nginx']
    }
    file { 'virtualhost':
        path      => '/etc/nginx/sites-available/default',
        (...)
    }
```

compliance.4labs.example

===== nginx.pp =====

```
class elastic_stack::nginx {
```

```
  package {'nginx':
    ensure => installed,
    before => Service['nginx']
  }
  file { 'virtualhost':
    path      => '/etc/nginx/sites-available/default',
    ensure => present,
    owner   => 'root',
    group  => 'root',
    mode   => '0644',
    source => 'puppet:///modules/elastic_stack/default',
    require => Package['nginx'],
    notify => Service['nginx']
  }
```

```
  service {'nginx':
    ensure => running,
    enable => true,
    require => File['virtualhost']
  }
```

```
}
```

Para o logstash, vamos criar alguns arquivos de configurações que serão utilizados pela aplicação.

Crie o arquivo filebeat-input.conf utilizando o modelo da pasta Aula 7.4:

```
# cp /vagrant/4525/arquivos/Aula\ 7.4/files/filebeat-input.conf
elastic_stack/files/
# cat elastic_stack/files/filebeat-input.conf
1 input {
    beats {
        port => 5443
        type => syslog
    }
}
```

compliance.4labs.example

===== filebeat-input.conf =====

```
input {
  beats {
    port => 5443
    type => syslog
  }
}
```


Crie o arquivo output-elasticsearch.conf utilizando o modelo da pasta Aula 7.4

```
# cp /vagrant/4525/arquivos/Aula\ 7.4/files/output-elasticsearch.conf
elastic_stack/files/
# cat elastic_stack/files/output-elasticsearch.conf
output {
1   elasticsearch { hosts => ["localhost:9200"]
      hosts => "localhost:9200"
      manage_template => false
      index => "%{[@metadata][beat]}-%{+YYYY.MM.dd}"
    }
}
```

compliance.4labs.example

===== output-elasticsearch.conf =====

```
output {
  elasticsearch { hosts => ["localhost:9200"]
    hosts => "localhost:9200"
    manage_template => false
    index => "%{[@metadata][beat]}-%{+YYYY.MM.dd}"
  }
}
```

Crie o arquivo syslog-filter.conf utilizando o modelo da pasta Aula 7.4:

```
# cp /vagrant/4525/arquivos/Aula\ 7.4/files/syslog-filter.conf
elastic_stack/files/
# cat elastic_stack/files/syslog-filter.conf
filter {
  if [type] == "syslog" {
    grok {
      match => { "message" => "%{SYSLOGTIMESTAMP:syslog_timestamp} %
{SYSLOGHOST:syslog_hostname} %{DATA:syslog_program}(?:\[%
{POSINT:syslog_pid}\])?: %{GREEDYDATA:syslog_message}" }
      add_field => [ "received_at", "%{@timestamp}" ]
      add_field => [ "received_from", "%{host}" ]
    }
  }
  (...)
}
```

compliance.4labs.example

===== syslog-filter.conf =====

```
filter {
  if [type] == "syslog" {
    grok {
      match => { "message" => "%{SYSLOGTIMESTAMP:syslog_timestamp} %
{SYSLOGHOST:syslog_hostname} %{DATA:syslog_program}(?:\[%{POSINT:syslog_pid}\])?: %
{GREEDYDATA:syslog_message}" }
      add_field => [ "received_at", "%{@timestamp}" ]
      add_field => [ "received_from", "%{host}" ]
    }
    date {
      match => [ "syslog_timestamp", "MMM d HH:mm:ss", "MMM dd HH:mm:ss" ]
    }
  }
}
```

Crie o manifest para a instalação do Logstash utilizando o modelo da pasta Aula 7.4:

```
# cp /vagrant/4525/arquivos/Aula\ 7.4/manifests/logstash.pp
elastic_stack/manifests/
# cat elastic_stack/manifests/logstash.pp
class elastic_stack::logstash {

  1   package {'logstash':
        ensure => installed,
        before => Service['logstash']
      }
      file { 'filebeat-input.conf':
        path      => '/etc/logstash/conf.d/filebeat-input.conf',
        (...)
      }
```

compliance.4labs.example

===== logstash.pp =====

```
class elastic_stack::logstash {
  package {'logstash':
    ensure => installed,
    before => Service['logstash']
  }
  file { 'filebeat-input.conf':
    path      => '/etc/logstash/conf.d/filebeat-input.conf',
    source => 'puppet:///modules/elastic_stack/filebeat-input.conf',
    require => Package['logstash'],
  }
  file { 'sysconfig-filter.conf':
    path      => '/etc/logstash/conf.d/sysconfig-filter.conf',
    source => 'puppet:///modules/elastic_stack/syslog-filter.conf',
    require => Package['logstash'],
  }
  file { 'output-elasticsearch.conf':
    path      => '/etc/logstash/conf.d/output-elasticsearch.conf',
    source => 'puppet:///modules/elastic_stack/output-elasticsearch.conf',
    require => Package['logstash'],
  }
  file_line {'JVM-logstash-XMS':
    path => '/etc/logstash/jvm.options',
    line => '-Xms512m',
    match => '-Xms1g',
    notify => Service['logstash']
  }
```

```
file_line {'JVM-logstash-XXM':
  path => '/etc/logstash/jvm.options',
  line => '-Xmx512m',
  match => '-Xmx1g',
  notify => Service['logstash']
}
service {'logstash':
  ensure => running,
  enable => true,
  require => [ File['output-elasticsearch.conf'],
    File['filebeat-input.conf'] ]
}
```

1

Crie o arquivo filebeat.yml que será responsável pela parametrização da aplicação:

```
# cp /vagrant/4525/arquivos/Aula\ 7.4/files/filebeat.yml
elastic_stack/files/
# cat elastic_stack/files/filebeat.yml
filebeat.inputs:
- type: log
  enabled: true
  paths:
    - /var/log/*.log
filebeat.config.modules:
  path: ${path.config}/modules.d/*.yml
  reload.enabled: false
output.logstash:
(...)
```

compliance.4labs.example

===== filebeat.yml =====

filebeat.inputs:

- type: log

enabled: true

paths:

- /var/log/*.log

filebeat.config.modules:

path: \${path.config}/modules.d/*.yml

reload.enabled: false

output.logstash:

hosts: ["log.4labs.example:5443"]

processors:

- add_host_metadata: ~

- add_cloud_metadata: ~

Crie o manifest para a instalação do Filebeat utilizando o modelo da pasta Aula 7.4:

```
# cp /vagrant/4525/arquivos/Aula\ 7.4/manifests/filebeat.pp
elastic_stack/manifests/
# cat elastic_stack/manifests/filebeat.pp
class elastic_stack::filebeat {

  1   package {'filebeat':
        ensure => installed,
        before => Service['filebeat']
      }
      file { 'filebeat.yml':
        path      => '/etc/filebeat/filebeat.yml',
        (...)
      }
```

compliance.4labs.example

===== filebeat.pp =====

```
class elastic_stack::filebeat {
```

```
  package {'filebeat':
    ensure => installed,
    before => Service['filebeat']
  }
```

```
  file { 'filebeat.yml':
    path      => '/etc/filebeat/filebeat.yml',
    source => 'puppet:///modules/elastic_stack/filebeat.yml',
    require => Package['filebeat'],
    notify => Service['filebeat']
  }
```

```
  service {'filebeat':
    ensure => running,
    enable => true,
    require => File['filebeat.yml']
  }
```

```
}
```

Após escrever todos os manifests, precisamos editar o `site.pp` para que sejam aplicados os manifests nos servidores corretos.

Atualize o arquivo `site.pp` utilizando o modelo da pasta Aula 7.4:

```
# cp /vagrant/4525/arquivos/Aula\ 7.4/site.pp
/etc/puppetlabs/code/environments/production/manifests
# cat /etc/puppetlabs/code/environments/production/manifests/site.pp
node 'log' {
    include elastic_stack::java8
    include elastic_stack::repo
    include elastic_stack::nginx
    include elastic_stack::elasticsearch
    include elastic_stack::kibana
    include elastic_stack::logstash
    (...)
}
```

compliance.4labs.example

===== site.pp =====

```
node 'log' {
    include elastic_stack::repo
        include elastic_stack::nginx
    include elastic_stack::elasticsearch
    include elastic_stack::kibana
    include elastic_stack::logstash
}
```

```
node 'compliance' {
    include inspec
    include elastic_stack::repo
    include elastic_stack::filebeat
}
```

```
node 'default' {
    include elastic_stack::repo
    include elastic_stack::filebeat
}}
```

Executando a instalação do Puppet Agent

Para terminar vamos aplicar a instalação do Puppet Agent em todas as máquinas do curso.

1

Acesse a máquina automation:
`# vagrant ssh automation`

2

Execute a playbook para a instalação do puppet-agent:
`# sudo ansible-playbook /etc/ansible/playbooks/puppet-agent.yml`

automation.4labs.example

Escrevendo Módulos no Puppet

Anotações

This image shows a blank sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.