02

Deployando uma aplicação web

Transcrição

Aprendemos a criar uma máquina do zero, colocar nela o Ubuntu, fazer o SSH e instala-la por meio de `apt-get update`, além dos pacotes de Java e Tomcat. Demos um IP para nossa máquina e acessamos o Tomcat via navegador, ele esta disponível na porta 8080, padrão do Tomcat. Conseguimos, ainda, instalar o Puppet para que, toda vez que eu o execute, ele faça o `apt-get update` e o install dos dois pacotes.

O próximo passo é pegar minha aplicação web e instalá-la no servidor. Em nosso máquina, as plicações são Java e possuem um arquivo formato .war, e esse arquivo será enviado para dentro do diretório `webapps` do meu Tomcat.

Temos uma aplicação baixada da internet de um projeto padrão do Vraptor chamado `musicjungle`. O arquivo .war deve ser copiado para o diretório onde temos os `manifests`. Dentro desse diretório, colocaremos o meu arquivo .war, o `vraptor-musicjungle.war`.

Agora que temos esse arquivo .war dentro do projeto, o que faremos é enviá-lo para o diretório do Tomcat. Para isso, precisaremos restartar o serviço, armazenado em `/etc/init.d`, onde ficam os serviços do Linux tradicionalmente.

Temos um serviço do Tomcat 7 chamado `tomcat7` dentro do diretório, comunicaremos essa informação ao Puppet. Se tiver algum problema com o `tomcat7`, não adianta copiamos o arquivo .war pra dentro do diretório do Tomcat, pois não teremos execução de qualquer maneira.

Nós definimos anteriormente que o pacote Tomcat 7 deve estar instalado. Agora estamos dizendo que existe um serviço chamado `tomcat7`, e esse serviço tem que estar rodando, caso não esteja, o Puppet não continuará a execução:

```
service { "tomcat7":
    ensure => running,
    enable => true,
    hasstatus => true,
    hasrestart => true,
    require => Package["tomcat7"]
}
```

O serviço `tomcat7` deve estar ativo e sendo executado, portanto `enable => true`. Também podemos saber o status do service via `hasstatus`. E o `hasrestart`, disponibiliza a opção de "restart" ou "stop" para o serviço. Por padrão, os serviços que a gente tem para Linux possuem esses recursos.

O Puppet verificará se o serviço está sendo executado, e caso não, irá reiniciá-lo. Para o serviço `tomcat7` estar rodando, primeiro temos de requerir (`require`) o pacote `tomcat7`.

Uma vez que o Puppet confirmou que serviço está operando normalmente, iremos copiar o arquivo .war. Para copiarmos esse arquivo, teremos de ir até o source isto é, o arquivo `/vagrant/manifests/vraptor-musicjungle.war`, e copiar para dentro do diretório `webapps` do Tomcat. Copiaremos o arquivo para `/var/lib/tomcat7/webapps/vraptor-musicjungle.war`.

```
file { "/var/lib/tomcat7/webapps/vraptor-musicjungle.war":
    source => "/vagrant/manifests/vraptor-musicjungle.war",
```

```
    owner => tomcat7,
    group => tomcat7,
    mode => 0644,
    require => Package["tomcat7"],
    notify => Service["tomcat7"]
}
```

Quando copiamos um arquivo no Linux, eu tenho que saber quem é o "dono" desse arquivo. Eu não queremos que o dono desse arquivo seja `root`, afinal o Puppet está sendo executado como `root`. Queremos executá-lo como usuário `tomcat7`.

Já existe um usuário `tomcat7` na máquina, porque o pacote do Tomcat já fez tudo isso para gente. Então, simplesmente declaramos que o grupo e o usuário é o `tomcat7`. O modo de acesso a esse arquivo é `0644`, de escrita,leitura, execução e assim por diante.

Depois de copiar esse arquivo, o Tomcat 7 deve ser reiniciado. É opcional – no caso do Tomcat, ele percebe que o arquivo war está lá – mas, para aprendermos como restartar o serviço, veremos passo a passo do procedimento.

Depois que o arquivo .war for instalado, solicitaremos que uma notificação para o serviço do Tomcat e, assim, ele ser reiniciado.

```
notify => Service["tomcat7"]
```

Estamos avisando para o Pupet que:

- Existe um serviço chamado 'tomcat7';
- O arquivo .war foi copiado de um diretório para outro.

Quando ativamos o `apply` nesse script (`sudo puppet apply /vagrant/manifests/web.pp`) podemos conferir que de fato o arquivo foi copiado de um diretório para outro, e o Tomcat foi reiniciado.

Como o arquivo foi copiado para dentro do Tomcat, ele já deve estar disponível na web.No navegador, acessaremos http://192.168.50.10:8080/vraptor-musicjungle (http://192.168.50.10:8080/vraptor-musicjungle). O MusciJunlge estará sendo executado.

Agora, todas as vezes que acionarmos `vagrant destroy` ou `vagrant up`, executaremos o `apply` e teremos a máquina configurada com o Tomcat e com a nossa aplicação web.

Resumindo: na linha de comando, copiamos o arquivo .war para o projeto, definimos o serviço, copiamos o arquivo de um diretório para dentro do nosso `webapps`. Por fim, Definimos o grupo do usuário e reiniciaremos o serviço.

```
exec { "apt-update":
    command => "/usr/bin/apt-get update"
}

package { [openjdk-7-j re", "tomcat7"]:
    ensure => installed,
    require => Exec["apt-update"]
}

service {"tomcat7":
```

```
    ensure => running,
    enable => true,
    hasstatus => true,
    hasrestart => true,
    require => Package["tomcat7"]
}

file {"/var/lib/tomcat7/webapps/vraptor-musicjungle.war":
    source => "/vagrant/manifests/vraptor-musicjungle.wae",
    owner => tomcat7,
    group => tomcat7
    mode => 0644,
    require => Package["tomcat7"],
    notify => Service["tomcat7"]
}
```

Agora faremos um SSH pra máquina e `apply` do nosso Puppet. Assim, conseguimos acessar o Music Jungle no navegador. Conseguimos nos cadastrar já que está sendo usado um banco de dados padrão em memória.

Até este ponto, vimos dois passos importantes: como criar máquinas virtuais e configurar o hardware. Depois, instalar o software mínimo como o sistema operacional da máquina virtual usando sempre o `vagrant` .

O `vagrant` , junto com virtual box, cria a máquina virtual.Depois, usamos o Puppet pra instalar o software. Assim como o Puppet, existem diversos outras tecnologias que servem pra instalar os softwares que a gente precisa configurar e gerenciar a configuração desse nosso ambiente. Se quisermos, acionamos o `vagrant reload` para reiniciar a máquina, do mesmo modo podemos utilizar `vegrant destroy` para elimininá-la.

Se reiniciarmos a máquina, não conseguiremos nos logar com o mesmo usuário. Como foi dito, o Musuc Jungle utiliza por padrão um banco de dados em memória, e quando reiniciamos a máquina esses dados são perdidos.

Para desenvolvimento, banco de dados em memória funciona perfeitamente pela agilidade, mas em produção não é uma boa ideia. O ideal nesse caso é utilizarmos o MySQL, mas continuarmos usando o banco de memória quando estamos no processo de desenvolvimento, dessa forma temos dois tipos de ambientes que precisam ser configurados.