jboss-domain-mode-scripts

Scripts para a automação de tarefas no JBoss EAP (em modo domain)

Paulo Jerônimo

Version 1.0, 2017-08-08 19:02:44 -03

Conteúdo

Sobre

O projeto

Este projeto, **jboss-domain-mode-scripts**, está disponível sob uma licença MIT no GitHub (em https://github.com/paulojeronimo/jboss-domain-mode-scripts). É da autoria de Paulo Jerônimo.

Seu intuito é ser utilizado para a **instalação rápida de um ambiente JBoss EAP rodando em modo domain**.

Neste momento, o uso desses scripts é suportado na versão 6.4 do JBoss EAP. Em breve, este projeto poderá ser extendido para a versão 7.0. Scripts similares a estes, também escritos por Paulo Jerônimo, já trazem o suporte a essa versão em outro projeto: jboss-scripts.

Diferente deste projeto, entretanto, o projeto **jboss-scritps** é voltado a utilização do JBoss EAP em ambiente de desenvolvimento, rodando em modo `standalone`.

A manutenção deste projeto é feita através do desenvolvimento de scripts codificados em Bash e/ou CLI.

O objetivo primário desses scripts é **facilitar a construção de ambientes de execução do JBoss EAP**, sejam eles de topologia simples ou complexa. Esses ambientes podem ser de desenvolvimento, homologação, produção ou qualquer outro.

Demonstrações de funcionamento dos scripts são realizadas neste projeto. Essas demonstrações se iniciam na construção de um ambiente simples, composto por um único **Domain Controller (DC)** e um **Host Controller (HC)**) (Demo 1: instalação de um DC e um HC). A seguir, conceitos de uso desses scripts são ampliados numa segunda demonstração, onde um um novo HC é adicionado. Por fim, a última demonstração apresenta como pode ser realizada a instalação de um DC adicional, atuando como backup do primeiro.

Concluindo, o maior benefício no uso deste projeto pode ser verificado quando é necessária a *construção de um ambiente grande, complexo, que envolve o uso de dois DCs e muitos HCs*.

Esta documentação

Objetivo

Esta documentação tem o objetivo de demonstrar o uso dos scripts do projeto. E, ela restringe-se a esse objetivo. Sendo assim, com relação ao público esperado na leitura deste documento, assume-se que:

1. Ele já tenha experiência no uso de comandos executados num shell Bash, em nível intermediário.
2. Ele já possua conhecimentos e experiência em JBoss EAP, particularmente em ambiente RHEL, em nível intermediário.

Para adquirir o conhecimento e a experiência necesssárias para um melhor entendimento deste documento, nos assuntos apontados acima, é recomendada a participação em treinamentos oficiais da Red Hat ou equivalentes.

Formato

Este documento é produzido na forma de um tutorial. Nele executamos demonstrações que explicam o funcionamento dos scripts deste projeto. Em algumas demonstrações, exercícios são propostos para fixar conteúdo já apresentado em momentos anteriores. As soluções para esses exercícios estão disponíveis ao final deste documento.

Criada através da ferramenta Asciidoctor (muito utilizada em vários projetos da Red Hat), esta documentação evolui da mesma maneira que o código deste projeto.

Ela também pode visualizada online, através de um acesso ao endereço https://paulojeronimo.github.io/jboss-domain-mode-scripts/docs/pt-br.

Os fontes em desenvolvimento deste documento estão no mesmo repositório que o projeto, no diretório docs/pt-br. Dessa forma, eles também são tratados como código e podem ter sua evolução visualizada através do GitHub. Assim como realizado no desenvolvimento de qualquer software gerenciado por esta ferramenta.

Paulo Jerônimo

Paulo Jerônimo é desenvolvedor de software. Também é consultor, e instrutor de treinamentos oficiais, na Red Hat. Em seu site, paulojeronimo.com, podem ser encontradas mais informações sobre suas atividades atuais.

Paulo Jerônimo também gosta de escrever documentos. Mas, utilizando linguagens de marcação leves. Ele se recusa, **veementemente**, a editar textos com quaisquer ferramentas da classe "Office". Seu editor de textos preferido, tanto na edição de códigos em linguagens de programação, quanto em textos em geral, é o Vim. Utilizando-se o Vim e o Asciidoctor podem ser enscritos documentos que vão de apresentações simples a grandes livros. Leia mais sobre isso no artigo "Asciidoc e Asciidoctor" (também escrito por Paulo Jerônimo).

Licença

Uma licença MIT é aplicada a este projeto. Segue o seu conteúdo:

```
The MIT License (MIT)

Copyright (c) 2017 Paulo Jerônimo

Permission is hereby granted, free of charge, to any person obtaining a copy of
this software and associated documentation files (the "Software"), to deal in
the Software without restriction, including without limitation the rights to
use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of
the Software, and to permit persons to whom the Software is furnished to do so,
subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

1. Motivação

A motivação principal para a construção deste projeto foi a necessidade de seu criador, Paulo Jerônimo, de não ficar se repetindo em tarefas de instalação de ambientes de execução do JBoss EAP, utilizando o sistema operacional Linux (RHEL).

A tarefa de instalação/manutenção do JBoss EAP precisava ser simplificada de forma tal que fosse possível, através da parametrização de poucos scripts de configuração (idealmente um), ser realizada a montagem de um ambiente em qualquer topologia.

Apesar de existir uma imagem Docker que facilita a execução do JBoss EAP, sua documentação descreve apenas como utilizá-la no modo `standalone`. Além disso, o uso dessa imagem Docker não é suportada pela Red Hat caso não exista uma subscrição. Havendo uma, porém, é necessário o uso da versão 7.0 (ou superior) do JBoss EAP. Finalmente, o seu ambiente de execução precisa ser um RHEL 7.2 ou superior.

Sendo assim, mesmo atendendo as condições necessárias para a utilização do JBoss EAP no Docker, ainda seria necessário extender a imagem para utilizá-la em modo domain. Portanto este projeto pode ser, também, uma base para a realização de um trabalho como esse.

2. Pré-requisitos

A primeira versão deste projeto contém scripts que podem ser executados somente pelo usuário `root`. O ambiente deve ser Linux (RHEL e/ou CentOS).

Uma cópia do instalador do Java Development Kit (JDK) para Linux deve ser disponibizada no diretório `installers`. O binário de instalação do JBoss (e seus patches) também devem ser copiados para esse diretório. Para mais informações, leia o `installers/README.adoc`.

Os arquivos que você deseja que sejam adicionados ao JBoss devem ser inseridos no diretório `configurations`. Esses arquivos incluem, por exemplo, binários de módulos (como drivers JDBC) e arquivos relativos a segurança (`*.pem`, `*.keystore`, etc). Também poderão ser incluídos neste diretório, patches que você deseja que sejam aplicados aos arquivos de configuração existentes no JBoss. Para mais informações, leia o `configurations/README.adoc`.

No diretório `scripts`, copie o arquivo scripts/config-sample para `config`. Em seguida, edite esse arquivo conforme tuas necessidades.

Se o arquivo `config` não for criado, o arquivo scripts/config-sample será utilizado em seu lugar.

3. Instalação completa (Pacotes, JDK, JBoss)

Essa forma de instalação, executada pelo script scripts/install, deve ser utilizada ao ser configurada uma nova máquina. Através dela, todos os pacotes necessários para o bom funcionamento do JBoss serão instalados. Além de pacotes necessários (por exemplo o JDK), pacotes opcionais podem ser instalados (exemplos: `tmux`, `lsof`, `vim`, etc). Verifique que pacotes que serão instalados editando o teu script de configuração (`config`).

Forma de execução:

```
./scripts/install
```

4. Instalação apenas do JBoss

Essa forma de instalação deve ser utilizada apenas para a reinstalação do JBoss. Ela só deve ser executada, após a instalação completa, caso seja necessária a reinstalação do JBoss. A execução do script scripts/jboss-install verificará a existência do usuário que executará o JBoss e, caso necessário, fará sua criação. Esse script também fará a cópia do diretório pré-configurado do JBoss (`JBOSS_INSTALLER`) para a sua localização de execução. Finalmente, ele ajustará os parâmetros necessários para realizar a execução do JBoss (como DC ou HC).

Forma de execução:

```
./scripts/jboss-install
```

5. Demonstrações de funcionamento

5.1. Pré-requisitos

5.1.1. Instalação de ferramentas

O teste dos scripts através das demonstrações a seguir exige a instalação de algumas ferramentas. São elas:

1. VirtualBox
2. Vagrant

Façamos a instalação dessas ferramentas antes de prosseguir. O VirtualBox será utilizado para a criação e o gerenciamento de máquinas virtuais. O Vagrant, por sua vez, será utilizado para provisionamento dessas máquinas.

1. O uso das ferramentas acima não é necessário caso já tenhamos acesso a máquinas (virtuais ou reais) que possam ser gerenciadas pelo usuário `root`. De fato, nesse caso, o uso dessas ferramentas é totalmente dispensável.
2. Os scripts que serão executados possuem independência completa do uso dessas ferramentas.
3. Nas demonstrações, apenas um provisionamento básico será realizado nas máquinas que serão criadas. Apesar de ser possível uma automação completa da instalação do JBoss através do provisionamento com o Vagrant, nas demonstrações a seguir, os passos que envolvem a instalação do JBoss serão realizados de maneira complemente manual. Dessa forma, espera-se que o leitor entenda o uso dos scripts e não como eles poderiam ser chamados através de um provisionamento automático pelo Vagrant.

5.1.2. Geração do pacote de instalação (jboss-installer.tar.gz)

Para cada construção de ambiente, um pacote (arquivo `jboss-installer.tar.gz`) precisará ser gerado. Esse pacote contém uma cópia de todos os arquivos necessários para a instalação do JBoss, em cada um dos servidores desse ambiente.

A geração do pacote deverá ser realizada pelo script jboss-installer.create. Esse script está disponível na raiz dos arquivos do projeto.

1. Veremos, em versões futuras dos scripts deste projeto, que esse pacote pode deixar de ser útil.
2. De fato, nessas novas versões, poderemos utilizar, por exemplo, o `rsync` ou `git` (preferencialmente). Serão esses aplicativos que farão a distribuição dos arquivos necessários, para cada servidor.

5.1.3. Configuração do arquivo /etc/hosts (na máquina HOST)

Como as demonstracões serão todas realizadas localmente, a partir de uma máquina HOST que executa máquinas virtuais, o arquivo `/etc/hosts` dessa máquina precisará ser configurado pelo script demos/scripts/configurar-hosts. Façamos isso, agora:

```
./demos/scripts/configurar-hosts
```

O comando acima deve ser executado na raiz dos arquivos do projeto. Da mesma forma, os próximos comandos, que serão realizados em demonstrações a partir do tópico seguinte, terão sempre a sua execução iniciada nesse diretório. Haverá um aviso quando essa não for a regra.

5.2. Demo 1: instalação de um DC e um HC

Nesta demonstração executaremos os passos necessários para a instalação de um DC (`dc1`) e de um HC (`hc1`). Nenhuma instância jboss (`server`) será criada. Mas, ao final desta demonstração, o `hc1` estará pronto e preparado para ser gerenciado pelo `dc1`.

5.2.1. Geração do pacote de instalação (jboss-installer.tar.gz)

Nosso primeiro passo, conforme explicado, será a construção do pacote `jboss-installer.tar.gz`. Para gerá-lo, executemos:

```
./jboss-installer.create
```

5.2.2. Criação/inicialização dos servidores

O Vagrant lerá o conteúdo de um arquivo `Vagrantfile` para criar e provisionar as duas máquinas desta demonstração. Dentro do diretório `demos`, vamos copiar o arquivo `1/Vagrantfile`:

```
cd demos
cp 1/Vagrantfile .
```

Esse é o conteúdo do arquivo `Vagrantfile` copiado nas linhas acima:

```
# -*- mode: ruby -*-
# vim: set ft=ruby ts=2 sw=2 expandtab:

required_plugins = %w( vagrant-vbguest )
required_plugins.each do |plugin|
  system "vagrant plugin install #{plugin}" unless Vagrant.has_plugin? plugin
end

Vagrant.configure(2) do |config|
  ENV['VAGRANT_DEFAULT_PROVIDER'] = 'virtualbox'

  config.vm.box = "boxcutter/centos73"
  config.vbguest.auto_update = false

  config.vm.define "dc1" do |dc1|
    dc1.vm.hostname="dc1"
    dc1.vm.network "private_network", ip: "172.17.6.81"
    dc1.vm.provision "shell", path: "scripts/instalar-vm", privileged: false
    dc1.vm.provider "virtualbox" do |v|
      v.memory = 1024
    end
  end

  config.vm.define "hc1" do |hc|
    hc.vm.hostname="hc1"
    hc.vm.network "private_network", ip: "172.17.6.82"
    hc.vm.provision "shell", path: "scripts/instalar-vm", privileged: false
    hc.vm.provider "virtualbox" do |v|
      v.memory = 2048
    end
  end
end
```

A linguagem utilizada para configurar um `Vagrantfile` é o Ruby. Sendo assim, um conhecimento mínimo nessa linguagem é o suficiente para a compreensão dos arquivos de configuração do Vagrant. Superpoderes (💵), entretanto, vão aparecendo conforme ampliamos o conhecimento (📚) nesta linguagem!

Agora vamos iniciar o Vagrant e aguardar que ele faça a criação dessas máquinas.

```
vagrant up
```

Podemos ir tomar um café ☕! ☺. Esse comando poderá demorar um pouquinho, dependendo da nossa velocidade de Internet.

5.2.3. Instalação e inicialização do DC

Voltemos ao diretório anterior (raiz do projeto):

```
cd ..
```

Vamos copiar o conteúdo do pacote (`jboss-installer.tar.gz`) para o servidor. Realizaremos isso chamando o jboss-installer.install da seguinte forma:

```
SSH_PORT=2222 SERVER=localhost ./jboss-installer.install
```

Na execução do script acima será solicitada a senha do usuário `root` (da máquina `dc1`). Informe: `vagrant`. Por padrão, as `boxes` de base criadas com o `vagrant` utilizam essa senha.

Normalmente, para acessar o shell da máquina `dc1`, utilizaríamos o comando `vagrant ssh dc1`. Entretanto, o que desejamos é executar comandos remotos com o usuário `root`. Aprofunde-se na leitura do jboss-installer.install. Você entenderá que o papel desse script é, simplesmente, copiar e extrair o conteúdo de `jboss-installer.tar.gz` num local padronizado (pela variável `PREFIX`).

Vamos instalar o JBoss:

```
ssh -p 2222 root@localhost /opt/rh/jboss-installer/scripts/install
```

A execução do comando acima demonstra como o scripts/install é executado. Esse script é responsável, como já citado, pela instalação completa do ambiente de execução do JBoss.

Façamos a inicialização do JBoss:

```
ssh -p 2222 root@localhost service jboss-as-domain start
```

O comando acima também poderia ser substituído pela execução do scripts/jboss-start. Este seria o comando equivalente:

```
ssh -p 2222 root@localhost /opt/rh/jboss-installers/scripts/jboss-start
```

Vejamos o log do JBoss:

```
ssh -p 2222 root@localhost tail -f /var/log/jboss-as/console.log
```

O comando acima também poderia ser substituído pela execução do scripts/jboss-tailf. Este seria o comando equivalente:

```
ssh -p 2222 root@localhost /opt/rh/jboss-installers/scripts/jboss-tailf
```

5.2.4. Instalação e inicialização do HC

Vamos abrir um outro terminal (shell Bash) para, agora, fazer a instalação do JBoss no hc1.

1. Um bom utilitário para multiplexar terminais é o tmux. Um material simples e rápido para aprender sua utilização é artigo "A tmux Crash Course".
2. Devemos nos lembrar que, ao abrir um novo terminal, precisaremos executar o próximo comando a partir da raiz do projeto.

Vamos fazer a cópia do conteúdo do pacote (jboss-installer.tar.gz) para esse servidor:

```
SSH_PORT=2200 SERVER=localhost ./jboss-installer.install
```

Observe que uma das diferenças entre a máquina dc1 e a máquina hc1 é o acesso. No caso, o hc1 é acessado pela porta 2200 enquanto que a dc1 é acessada pela porta 2222.

Vamos utilizar uma "técnica" para a "Minimização de dedadas" (no teclado ☺):

```
scripts=/opt/rh/jboss-installer/scripts
alias hc1ssh='ssh -p 2200 root@localhost'
```

Instalemos o JBoss:

```
hc1ssh $scripts/install
```

Façamos sua inicialização:

```
hc1ssh $scripts/jboss-start
```

Podemos verificar, no terminal aberto para a visualização do log do JBoss no dc1, que o hc1 deverá ter sido registrado.

E vejamos o seu log:

```
hc1ssh $scripts/jboss-tailf
```

5.2.5. Verificação de funcionamento do ambiente

Vamos abrir o browser em http://dc1:9990. Façamos o login com o usuário `admin` e a senha `jbAdmin@123!`. Precisamos acessar a aba `Domain` e verificar se o `hc1` está registrado na lista de hosts. Em caso positivo, concluímos com sucesso esta demonstração.

5.3. Demo 2: adição de um segundo HC e dois servers

Nesta demonstração, alteraremos o ambiente para a adição de um novo HC (`hc2`). Além disso, nesse ambiente, duas instâncias jboss serão iniciadas. Cada instância será executada em seu próprio HC. Essas instâncias terão o mesmo nome (`app-1`) e pertencerão ao mesmo `Server Group`: `app-1-group`.

Ainda exploraremos as limitações relativas a uma montagem de ambiente utilizando os scripts nesta versão. Contudo, já neste momento, é bom saber que somente através do uso desses scripts, e se não levássemos em conta a possibilidade de uso do console de gerenciamento do JBoss, ou a sua capacidade de execução de scripts CLI, estaríamos muito, muito limitados.

O motivo disso é que, basicamente, os scripts dessa versão são excelentes para a instalação de infraestruturas estáticas (DCs e HCs). Porém, para lidar com a dinamicidade que envolve a criação de grupos de servidores e servidores, muitas vezes em direfentes `profiles`, esses scripts não conseguem oferecer o auxílio necessário. Precisaria ser inserido, nesses scripts, um mecanismo que nos possibilitasse executar scripts CLI de maneira mais direta e sistematica. Mais a frente, veremos como isso pode ser feito.

5.3.1. Adição do segundo HC (hc2) - (utilizando patch e diff)

Vamos utilizar o comando patch. Ele recebe como entrada, um arquivo com a extensão `.patch`. Um arquivo `patch` mostra alterações entre duas versões de um mesmo arquivo. Ele informa que linhas devem ser adicionadas (`+`) ou removidas (`-`) no arquivo em que o `patch` será aplicado.

Este é o conteúdo do arquivo Vagrantfile.patch:

```
--- 1/Vagrantfile   2017-07-25 10:39:30.000000000 -0300
+++ Vagrantfile 2017-07-26 08:11:11.000000000 -0300
@@ -21,12 +21,14 @@
     end
   end

-  config.vm.define "hc1" do |hc|
-    hc.vm.hostname="hc1"
-    hc.vm.network "private_network", ip: "172.17.6.82"
-    hc.vm.provision "shell", path: "scripts/instalar-vm", privileged: false
-    hc.vm.provider "virtualbox" do |v|
-      v.memory = 2048
+  (1..2).each do |i|
+    config.vm.define "hc#{i}" do |hc|
+      hc.vm.hostname="hc#{i}"
+      hc.vm.network "private_network", ip: "172.17.6.8#{i+2}"
+      hc.vm.provision "shell", path: "scripts/instalar-vm", privileged: false
+      hc.vm.provider "virtualbox" do |v|
+        v.memory = 2048
+      end
     end
   end
 end
```

Fazendo sua leitura identificamos que as linhas de configuração do `hc1` foram removidas e, no lugar delas, um "loop" (com duas iterações) entrou. Esse "loop" (`(1..2).each |v|`) fará, então, a criação do `hc2`.

Para aplicar esse `patch`, modificando o conteúdo do arquivo `Vagrantfile`, basta executar o comando abaixo:

```
cd demos
patch Vagrantfile < 2/Vagrantfile.patch
```

Em relação a demo 1, o `Vagrantfile` que acabamos de modificar com o `patch` faz, conforme explicado, a adição do `hc2`.

Por curiosidade, você poderia ver como gerar um `patch` através do comando comando diff:

```
diff -uNr 1/Vagrantfile Vagrantfile
```

Em seguida, para termos certeza de que, realmente, o conteúdo deste `patch` é o que está no arquivo `2/Vagrantfile.patch`, poderíamos utilizar o Vim, através deste comando:

```
vim -d <(!!) 2/Vagrantfile.patch
```

A saída do comando `diff` tem, aproximadamente (mudando-se um `timestamp`), o mesmo conteúdo do arquivo demos/2/Vagrantfile.patch. Dessa forma, se você ainda não sabia como gerar um `patch` (e aplicá-lo), este tópico acaba de lhe ensinar como fazer isso (através do `diff`). O que os scripts deste projeto executam *"under the hood"*, desde a primeira demonstração, é uma aplicação sistemática de `patches`, de forma a fazer os ajustes necessários para colocar o JBoss em execução, de acordo com certos parâmetros.

Após a aplicação do `patch`, vamos solicitar ao Vagrant que inicie as máquinas:

```
vagrant up
```

Quando na execução do comando acima, o Vagrant verificará que as máquinas `dc1` e `hc1` já estão em execução e foram provisionadas. Portanto, para elas, nada será realizado. Mas, seguindo em frente, o Vagrant verificará que a máquina `hc2` precisará ser criada. E fará a sua criação e provisionamento.

5.3.2. Criação de arquivos de configuração

Para esta demonstração precisamos, em relação a demo 1, de três (3) arquivos adicionais. Eles são apresentados na saída do comando a seguir (executado no diretório `demos`):

```
$ tree 2/{configurations,scripts}
2/configurations
`-- jboss-eap-6.4
    `-- domain
        `-- configuration
            |-- domain.xml.patch
            `-- host-slave.xml.no-backup.srv1.patch
2/scripts
`-- config

3 directories, 3 files
```

Só poderemos executar esse comando quando terminarmos os passos que precisamos seguir a partir de agora. Portanto, voltaremos a ele ao final desse tópico.

Esses arquivos serão utilizados pelos scripts deste projeto, no processo de instalação do JBoss. Mas, eles ainda não existem. Então, vamos criá-los.

scripts/config

O arquivo `scripts/config` nós já conhecemos. Quer dizer, ainda não tão detalhadamente. Mas, podemos ler mais a respeito dele vendo o conteúdo do scripts/config-sample.

O `config` é o arquivo citado no tópico "Pré-requisitos". É importante saber que, na demo 1, não

fizemos nenhuma configuração. Não criamos o arquivo `config`. Mesmo assim, as coisas funcionaram: o `hc1` conseguiu se conectar ao `dc1`. Isso quer dizer, como explicado, que foi utilizado o scripts/config-sample.

Nesta demonstração, entretanto, queremos ir um pouco além. Queremos configurar um novo `Server Group` e, também, duas instâncias executando em dois HCs diferentes. Para isso, precisamos criar o nosso arquivo `config`. Nele, configuraremos que tanto `hc1` quanto `hc2` ajustarão o valor `srv1` para a variável `JBOSS_SERVER_CONFIG`. Fazendo essa configuração, quando o scripts/jboss-configure for chamado (pelo scripts/jboss-install), ele tentará aplicar um `patch` chamado `host-slave.xml.no-backup.srv1.patch` ao arquivo `host-config.xml`.

Como podemos criar o arquivo `config`? Simples! A maneira mais fácil seria copiar o scripts/config-sample para `scripts/config` e utilizá-lo como base na nossa edição. Nele, dentro do `case` da linha 72, as seguintes linhas poderiam ser inseridas:

demos/2/scripts/config

```
  # If host is hc1 or hc2, JBOSS_SERVER_CONFIG will be srv1
  hc[1,2])
    JBOSS_SERVER_CONFIG=srv1
    ;;
```

Contudo, já sabemos como funciona um `patch`! Então, ao invés de editarmos o arquivo `config`, faremos a sua cópia a partir do scripts/config-sample, e aplicaremos o `patch` config.patch. Para isso, executemos os seguintes comandos:

```
mkdir -p 2/scripts
cp ../scripts/config.sample 2/scripts/config
(cd 2/scripts; patch config < ../config.patch)
```

A aplicação desse `patch` fará o ajuste informado nas linhas acima. Obviamente, o conteúdo deste `patch` apresenta contém essas linhas. Esse é o conteúdo do `config.patch`:

```
--- config.sample   2017-07-27 16:14:40.000000000 -0300
+++ config  2017-07-27 16:23:40.000000000 -0300
@@ -76,6 +76,10 @@
      JBOSS_TYPE=master
      ;;

+  # If host is hc1 or hc2, JBOSS_SERVER_CONFIG will be srv1
+  hc[1,2])
+    JBOSS_SERVER_CONFIG=srv1
+    ;;
 esac

 # Host config file is JBOSS_TYPE dependant
```

Poderíamos ser redundantes (mas seguros) fazendo uma dupa checagem do conteúdo do arquivo arquivo gerado (`2/scripts/config`). Se guiséssemos fazer isso, poderíamos executar o seguinte comando: `grep JBOSS_SERVER_CONFIG -A 4 2/scripts/config`.

configurations/jboss-eap-6.4/domain/configurations/domain.xml.patch

Para criar um `patch`, como já vimos, precisamos do arquivo original. No processo de instalação do JBoss, executado por este projeto, a versão que será instalada é uma em que os `patches` do JBoss já estão aplicados. "Patches do JBoss" são os arquivos binários, configurados no arquivo `config`, como nas linhas apresentadas pela saída do comando `"grep -A 4 JBOSS_PATCHES 2/scripts/config"`:

```
JBOSS_PATCHES=(
  jboss-eap-6.4.9-patch.zip
  jboss-eap-6.4.16-patch.zip
)
```

A versão do JBoss que será instalada, após seus `patches` serem aplicados, estará disponível em `/opt/rh/jboss-installer/installers/jboss-eap-6.4`.

O diretório `/opt/rh/jboss-installer/installers/jboss-eap-6.4` não é o diretório em que o JBoss final será executado! Esse diretório é criado pelo scripts/jboss-extract, caso ele não exista. Essa criação é realizada pela extração do `zip` de instalação do JBoss. Em seguida, nesse diretório, são aplicados os `patches` de atualização do JBoss. Após a aplicação dos `patches`, o scripts/jboss-extract também mescla o conteúdo disponível no diretório `configurations`, com o conteúdo extraído.

Dessa forma, podemos copiar o arquivo original que desejamos configurar (`domain.xml`), com os seguintes comandos:

```
b=2/configurations
o=/opt/rh/jboss-installer/installers
d=jboss-eap-6.4/domain/configuration
mkdir -p $b/$d && cd $_
scp -P 2200 root@localhost:$o/$d/domain.xml .
```

Antes de editar o `domain.xml` com as alterações que queremos, façamos uma cópia dele para `domain.xml.original`:

```
cp domain.xml{,.original}
```

Agora, podemos editar o arquivo com as alterações que desejamos. No caso, desejamos eliminar os `server-groups` `default`: `main-server-group` e `other-server-group`. Além disso, queremos criar o `server-group` `app1-server-group`, utilizando o profile `default`.

Utilizando o Vim, vamos editar o arquivo `domain.xml`, alterá-lo de acordo com as mudanças que

desejamos, e salvá-lo. Mas, ao invés de fazer isso de maneira interativa (entrando comandos no editor), vamos executar o script 2/domain.xml.vim para fazer as mudança que queremos:

```
vim -c "source ../../../../domain.xml.vim" domain.xml
```

O conteúdo desse `domain.xml.vim` é extremamente simples, e didático, para quem já tem conhecimentos básicos no Vim. Veja:

```
1400,1412d
1400
i
        <server-group name="app-1-server-group" profile="default">
            <socket-binding-group ref="standard-sockets"/>
.
x
```

Pela sua leitura, observamos o seguinte:

1. 13 linhas serão excluídas (da linha 1400 a 1412).
2. Será inserido, na linha 1400, o conteúdo apresentado.
3. O arquivo será salvo e a execução do Vim será encerrada.

Faremos a criação do `domain.xml.patch` executando:

```
diff -uNr domain.xml{.original,} > domain.xml.patch
```

Finalmente, removeremos os arquivos `domain.xml` e `domain.xml.original` pois eles não são utilizados pelos scripts deste projeto:

```
rm -f domain.xml{.original,}
```

configurations/jboss-eap-6.4/domain/configuration/host-slave.xml.no-backup.patch

Voltaremos ao diretório anterior (`demos`) para, agora, buscar outro arquivo de configuração para o qual precisaremos criar um `patch`. Estamos falando do arquivo `host-slave.xml`. Executemos:

```
cd -
cd $b/$d && scp -P 2200 root@localhost:$o/$d/host-slave.xml .
```

Antes de editar esse arquivo, vamos salvar o seu conteúdo original:

```
cp host-slave.xml{,.original}
```

Para que o scripts/jboss-configure consiga fazer o seu trabalho, configurar o `hc2` da forma necessária, precisamos editar esse arquivo. Faremos isso, através do script `2/host-slave.xml.vim`, executando o seguinte comando:

```
vim -c "source ../../../../host-slave.xml.vim" host-slave.xml
```

O conteúdo do arquivo `host-slave.xml.vim`, da mesma forma que o `domain.xml.vim`, também é muito simples. Ele indica, como vimos, as mudanças que devem ser realizadas. Seu conteúdo (com a apresentação de um número de linha) é o seguinte:

```
 1 10d
 2 i
 3                     <secret value="JBOSS_ADMIN_PASSWORD_BASE64"/>
 4 .
 5 58d
 6 i
 7         <remote host="${jboss.domain.master.address}" port
="${jboss.domain.master.port:9999}" security-realm="ManagementRealm" username
="JBOSS_ADMIN_USER"/>
 8 .
 9 86,91d
10 i
11         <server name="app-1" group="app-1-server-group"/>
12 .
13 x
```

Pela sua leitura, observamos o seguinte:

1. A linha 10 é alterada de forma que, para o atributo `value` do elemento `secret`, agora temos o valor `JBOSS_ADMIN_PASSWORD_BASE64`.
2. A linha 58 é alterada adicionando-se o atributo `username`, cujo valor é `JBOSS_ADMIN_USER`.
3. A definição dos grupos `default`, entre as linhas 86 e 91 é removida. Em seu lugar é feita a configuração do servidor `app-1`.

Como citado no objetivo deste documento, está fora de seu escopo explicar como é realizada a configuração para que um HC se conecte a um DC. Administradores experientes em JBoss EAP aprendem isso num curso básico. Contudo, é necessário saber que os valores para as variáveis `JBOSS_ADMIN_PASSWORD_BASE64` e `JBOSS_ADMIN_USER` serão ajustados logo após a execução do scripts/jboss-configure. Dessa forma, os valores dessas variáveis precisam estar cofigurados no arquivo `config`.

Para encerrar a construção dos arquivos de configurações que precisamos, no intuito de dar continuidade a esta demonstração, vamos gerar o `patch` (e excluir os arquivos desnecessários após isso) executando os comandos a seguir:

```
diff -uNr host-slave.xml{.original,} > host-slave.xml.no-backup.srv1.patch
rm -f host-slave.{.original,}
```

Um detalhe importante de ser explicado é o nome que demos a esse `patch`: `host-slave.xml.no-backup.srv1.patch`. É necessário que ele tenha esse nome. Isso porque o scripts/jboss-configure executará a seguinte lógica:

```
f=domain/configuration/$JBOSS_HOST_CONFIG
f_patch=$f
if [ "$JBOSS_HOST_CONFIG" = "host-slave.xml" ]
then
  [ "$JBOSS_DOMAIN_BACKUP_ADDRESS" ] || f_patch=$f_patch.no-backup
fi
[ "$JBOSS_SERVER_CONFIG" ] && f_patch=$f_patch.$JBOSS_SERVER_CONFIG
patch $f $f_patch
```

Como configuramos (no config) que `JBOSS_SERVER_CONFIG` terá o valor `srv1` e, como ainda não especificamos um valor para `JBOSS_DOMAIN_BACKUP_ADDRESS` (veremos isso no próximo tópico), então o nome do arquivo precisará seguir o padrão estipulado (acima) e deverá ser o que informamos.

Verificação

Para ter certeza que tudo está em seu devido lugar, executemos os seguintes comandos:

```
cd -
tree 2/{configurations,scripts}
```

Esperamos que o resultado desse último comando seja apresentado no início deste tópico.

5.3.3. Geração do pacote de instalação (jboss-installer.tar.gz)

Voltemos a raiz do projeto:

```
cd ..
```

A inclusão do parâmetro `DEMO=2`, antes da execução do script `jboss-installer.create`, faz com que sejam copiados, para o pacote `jboss-installer.tar.gz`, o conteúdo dos scripts e das configurações que estão disponíveis no diretório `demos/2`. Executemos:

```
DEMO=2 ./jboss-installer.create
```

O resultado da execução acima é um pacote que, além de conter os script e configurações padrão, também possuirá o conteúdo de `demos/2/scripts` e `demos/2/configurations`. Verifiquemos isso executando o comando abaixo:

```
tar tvf jboss-installer.tar.gz
```

5.3.4. Finalização do JBoss em dc1 e hc1

Vamos parar as instâncias JBoss já em execução em dc1 e em hc1. Executemos o seguinte comando:

```
for p in 22{00,22}; do ssh -p $p root@localhost /opt/rh/jboss-installer/scripts/jboss-stop; done
```

Recordando o que aprendemos ao final da demo 1: além do scripts/jboss-stop, também há comandos para as tarefas de inicialização, otenção do estado de execução, e visualização do log do JBoss: scripts/jboss-start, scripts/jboss-status, scripts/jboss-tailf.

5.3.5. Exercício 1: substituição do instalador, reinstalação e inicialização do DC (dc1)

Repita os mesmos passos que já foram executados (na demo-1) para a instalação e inicialização do DC.

Requisitos:

1. Ao invés de gastar os dedos da forma como realizado em "Instalação e inicialização do DC", seja `smart` e utilize o mecanismo de "minimização de dedadas" apresentado em "Instalação e inicialização do HC".
2. Utilize o comando `sshpass` (se não souber o que ele faz, busque no Google!).

Veja a solução no tópico "Soluções de exercícios" em "Demo 2, exercício 1".

5.3.6. Exercício 2: reinstalação do hc1 e instalação do hc2

Repita, para cada HC, os mesmo passos que já foram executados (na Demo 1) para a instalação e inicialização do hc1.

1. Da mesma forma que realizado no exercício anterior, você precisará substituir o instalador do hc1.
2. Ao serem iniciados os passos para a realização da instalação do hc2 deve ser observado que a porta utilizada é a 2201.

Veja a solução no tópico "Soluções de exercícios" em "Demo 2, exercício 2".

5.3.7. Verificação de funcionamento do ambiente

Abra o browser em http://dc1:9990. Logue-se com o usuário `admin` e a senha `jbAdmin@123!`. Acesse a

aba `Domain` e verifique que tanto o `hc1` quanto o `hc2` estão registrados na lista de hosts. Verifique também, na aba topologia, que há duas instâncias de `app-1` sendo executadas.

5.4. Demo 3: Instalação de um DC adicional (backup)

5.4.1. O que é um DC de backup?

Um DC tem o papel primário de fazer o gerenciamento de um HC (ou vários). Quando ele fica indisponível, o funcionamento das aplicações do domínio não é impactado. Entretanto, o administrador do domínio fica incapacitado de gerenciar instâncias. Tarefas de administração e configurações dos servidores são impactadas na ausência do DC. Sendo assim, é importante que exista um mecanismo de alta disponibilidade para um DC.

O JBoss EAP provê a possibilidade de se instalar e gerenciar um DC de backup. Criando-se um DC de backup, é eliminada a impossibilidade de se administrar o domínio na ausência do DC primário (master). Na documentação do JBoss EAP, o tópico "About Domain Controller Discovery and Failover" trata da configuração de um DC de backup. Nos próximos tópicos desta documentação, é apresentado, passo a passo, como configurar um DC de backup. Também, é simulada uma indisponbilidade do DC primário e o que deve ser feito para colocar um DC de backup em ação.

5.4.2. Configuração

A configuração do `Vagrantfile` para a adição do `dc2` pode ser realizada pela aplicação do `patch` demos/3/Vagrantfile.

Este é o seu conteúdo:

```
--- Vagrantfile 2017-07-26 08:51:18.000000000 -0300
+++ 3/Vagrantfile   2017-07-25 10:39:30.000000000 -0300
@@ -12,16 +12,16 @@
   config.vm.box = "boxcutter/centos73"
   config.vbguest.auto_update = false

-  config.vm.define "dc1" do |dc1|
-    dc1.vm.hostname="dc1"
-    dc1.vm.network "private_network", ip: "172.17.6.81"
-    dc1.vm.provision "shell", path: "scripts/instalar-vm", privileged: false
-    dc1.vm.provider "virtualbox" do |v|
-      v.memory = 1024
+  (1..2).each do |i|
+    config.vm.define "dc#{i}" do |dc|
+      dc.vm.hostname="dc#{i}"
+      dc.vm.network "private_network", ip: "172.17.6.8#{i}"
+      dc.vm.provision "shell", path: "scripts/instalar-vm", privileged: false
+      dc.vm.provider "virtualbox" do |v|
+        v.memory = 1024
+      end
     end
-  end

-  (1..2).each do |i|
     config.vm.define "hc#{i}" do |hc|
       hc.vm.hostname="hc#{i}"
       hc.vm.network "private_network", ip: "172.17.6.8#{i+2}"
```

Façamos a aplicação deste `patch`:

```
cd demos
patch Vagrantfile < 3/Vagrantfile.patch
```

Precisamos, também, gerar o `scripts/config` dessa demo, baseando-nos na configuração realizada para a demo anterior. O `patch` 3/config.patch pode ser utilizado para isso. Seu conteúdo é o seguinte:

```
--- ../2/scripts/config 2017-08-03 10:56:40.000000000 -0300
+++ scripts/config  2017-08-03 11:14:15.000000000 -0300
@@ -60,6 +60,9 @@
 # JBoss domain (master) address (ip or hostname)
 JBOSS_DOMAIN_MASTER_ADDRESS=dc1

+# JBoss domain (backup) address (ip or hostname)
+JBOSS_DOMAIN_BACKUP_ADDRESS=dc2
+
 # JBoss admin username (to access console)
 JBOSS_ADMIN_USER=admin
 # JBoss admin password
@@ -72,7 +75,7 @@
 case $JBOSS_HOST in

   # If host is master, JBOSS_TYPE is master
-  $JBOSS_DOMAIN_MASTER_ADDRESS)
+  $JBOSS_DOMAIN_MASTER_ADDRESS|$JBOSS_DOMAIN_BACKUP_ADDRESS)
     JBOSS_TYPE=master
     ;;
```

A geração do arquivo `3/scripts/config` pode ser realizada da seguinte forma:

```
cp 2/scripts/config 3/scripts/
(cd 3/scripts; patch config < config.patch)
```

Podemos executar o comando abaixo para visualizar, através do Vim, que diferenças há entre o `config` da demonstração 2 e o `config` que acabamos de criar:

```
vim -d {2,3}/scripts/config
```

Pela visualização das diferenças, apresentadas pelo Vim, notamos que foram adicionadas informações específicas para o `dc2`. Mais a frente, nas explicações sobre o arquivo de configuração, obteremos detalhes a respeito disso. Nossa preocupação, até o final desse tópico sobre demonstrações, será apenas a de saber como executar os scripts.

5.4.3. Geração do pacote de instalação (jboss-installer.tar.gz)

Agora que já temos o arquivo `scripts/config`, assim como outros arquivos no diretório `configurations`, podemos gerar o pacote de instalação:

```
DEMO=3 ./jboss-installer.create
```

5.4.4. Inicialização do segundo DC (dc2)

5.4.5. Instalação do dc2

```
cd ..
alias dc2ssh='ssh -p 2202 root@localhost'
dc2ssh $scripts/install
```

5.4.6. Referências extras

- High Availability of the Domain Controller in JBoss EAP 6
- JBoss EAP 6.3 Domain controller failover demo

6. Scripts

6.1. jboss-installer.create

```
#!/bin/bash
set +x
set -e

BASE_DIR=`cd "$(dirname "$0")"; pwd`
INSTALLER=${INSTALLER:-jboss-installer.tar.gz}
PREFIX=${PREFIX:-/opt/rh/jboss-installer}
DEMO=${DEMO:-1}

trap "{ rm -rf merged; }" EXIT
trap "{ rm -f "$INSTALLER"; }" INT

[[ $OSTYPE =~ darwin.* ]] && tar=gtar || tar=tar

cd "$BASE_DIR"

[ -d "demos/$DEMO" ] || {
  echo "DEMO directory (demos/$DEMO) does not exists!"
  exit 1
}

rsync -a scripts configurations merged
[ -d "demos/$DEMO/scripts" ] && \
  rsync -qa demos/$DEMO/{scripts,configurations} merged/

echo "Creating $INSTALLER ..."
find . -type f \( \
  -path './installers/*' -o \
  -path './merged/*' \
  \) \( ! -name README.adoc -o -name "$config" \) -print0 | \
$tar cvfz $INSTALLER \
--transform "s,^\./merged,.," \
--transform "s,^\.,$PREFIX," \
--null -T -
```

- Download: https://paulojeronimo.github.io/jboss-domain-mode-scripts/jboss-installer.create

6.2. jboss-installer.install

```
#!/bin/bash
set +x
set -e

BASE_DIR=`cd "$(dirname "$0")"; pwd`
INSTALLER=${INSTALLER:-jboss-installer.tar.gz}
PREFIX=${PREFIX:-/opt/rh/jboss-installer}
SSH_USER=${SSH_USER:-root}
SSH_PORT=${SSH_PORT:-22}

ssh() {
  local ssh=`which ssh`
  local params="-p $SSH_PORT $SSH_USER@$SERVER $@"

  [ "$SSHPASS" ] && sshpass -e $ssh $params || $ssh $params
}

if ! [ "$SERVER" ]
then
  echo "Usage: SERVER=<server> $0"
  exit 1
fi

if ! [ -f "$INSTALLER" ]
then
  echo "INSTALLER ($INSTALLER) not found!"
  exit 1
fi

SCRIPT=$(cat << EOF
echo "I'm at \$HOSTNAME ..." ;
[ -d "$PREFIX" ] && { echo "Removing existing $PREFIX dir ..." ; rm -rf $PREFIX ; };
echo "Extracting files from pipe to / ..." ;
tar xvfmPz - -C /
EOF
)

cd "$BASE_DIR"
cat $INSTALLER | ssh $SCRIPT
```

- Download: https://paulojeronimo.github.io/jboss-domain-mode-scripts/jboss-installer.install

6.3. demos/scripts/configurar-hosts

```
#!/bin/bash
set +x

configurar() {
    local f=/etc/hosts

    grep -q "^$1.*$2" $f || { echo -e "$1\t$2" | sudo tee -a $f > /dev/null; }
}

if [ "$1" -a "$2" ]
then
  configurar $1 $2
else
  configurar 172.17.6.81 dc1
  configurar 172.17.6.82 dc2
  configurar 172.17.6.83 hc1
  configurar 172.17.6.84 hc2
fi
```

- Download: https://paulojeronimo.github.io/jboss-domain-mode-scripts/demos/scripts/configurar-hosts

6.4. scripts/common

```
#!/bin/bash
set +x
set -e

source "$BASE_DIR"/functions

LINUX_NEEDED=${LINUX_NEEDED:-true}
if $LINUX_NEEDED && ! [[ $OSTYPE =~ linux* ]]
then
  echo "This script needs to be executed on Linux!"
  exit 1
fi

ROOT_NEEDED=${ROOT_NEEDED:-true}
if $ROOT_NEEDED && (( `id -u` != 0 ))
then
  echo "You must run this script as root (or with sudo)!"
  exit 1
fi

config="$BASE_DIR"/config
load_configuration_warned=${load_configuration_warned:-false}
[ -r "$config" ] || config=$config.sample

source "$config"

if $VERBOSE
then
  $load_configuration_warned || {
    echo "Loading JBoss configurations from \"$config\" ..."
    export load_configuration_warned=true
  }
fi
```

- Download: https://paulojeronimo.github.io/jboss-domain-mode-scripts/scripts/common
- Dependências:
  - scripts/functions

6.5. scripts/config-sample

```
#!/bin/bash

# The INSTALLERS_DIR must be relative to this directory
# It have to contains binaries like jboss-eap-6.4.0.zip, jboss-eap-6.4.*.patch.zip
INSTALLERS_DIR=../installers

# The CONFIGURATIONS_DIR must be relative to this directory
# It have to contains patch and additional files that will be added to JBOSS_DIR
CONFIGURATIONS_DIR=../configurations

# Scripts will show warnings?
VERBOSE=false

# JDK Installer is a RPM binary file
JDK_INSTALLER=jdk-8u144-linux-x64.rpm

# Binay file to extract JBoss (PATH is relative to INSTALLERS_DIR)
# These files are the original downloaded from https://access.redhat.com
JBOSS_INSTALLER_ZIP=jboss-eap-6.4.0.zip
JBOSS_PATCHES=(
  jboss-eap-6.4.9-patch.zip
  jboss-eap-6.4.16-patch.zip
)

# JBoss directory when extracted
JBOSS_DIR=jboss-eap-6.4

# User (operational system) that runs JBoss
JBOSS_USER=jboss
JBOSS_USER_PASSWORD='jb@3!2_'
# Group (operational system) that runs JBoss
JBOSS_GROUP=jboss

# Directory that will contains $JBOSS_DIR
JBOSS_INSTALL_DIR=/opt

# Link (bellow JBOSS_INSTALL_DIR) that will points to $JBOSS_DIR
JBOSS_LINK=jboss

# Full path to JBoss
JBOSS_HOME=$JBOSS_INSTALL_DIR/$JBOSS_LINK

# JBoss console log file
JBOSS_CONSOLE_LOG=/var/log/jboss-as/console.log

# Hostname of the machine that runs JBoss
JBOSS_HOST=`hostname -s`
```

```
# JBoss bind address (ip or hostname)
case "$OSTYPE" in
  # In this case (linux), it is configured with IP address of the second interface
  linux*) JBOSS_BIND_ADDRESS=`hostname -I | cut -d ' ' -f 2`;;
  # We put a macOS alternative just to permit a jboss-copy-patches script execution
  darwin*) JBOSS_BIND_ADDRESS=localhost;;
esac

# JBoss bind address management (ip or hostname)
JBOSS_BIND_ADDRESS_MANAGEMENT=$JBOSS_BIND_ADDRESS

# JBoss domain (master) address (ip or hostname)
JBOSS_DOMAIN_MASTER_ADDRESS=dc1

# JBoss admin username (to access console)
JBOSS_ADMIN_USER=admin
# JBoss admin password
JBOSS_ADMIN_PASSWORD='jbAdmin@123!'

# Default JBoss type
JBOSS_TYPE=slave

# Variable configurations depending on $JBOSS_HOST
case $JBOSS_HOST in

  # If host is master, JBOSS_TYPE is master
  $JBOSS_DOMAIN_MASTER_ADDRESS)
    JBOSS_TYPE=master
    ;;

esac

# Host config file is JBOSS_TYPE dependant
JBOSS_HOST_CONFIG=host-${JBOSS_TYPE}.xml
```

- Download: https://paulojeronimo.github.io/jboss-domain-mode-scripts/scripts/config.sample

6.6. scripts/functions

```
#!/bin/bash

patch() {
  local cmd=`which patch`
  local f=$1
  local f_patch=$2.patch
  local original=$f.original

  if [ ! -f "$original" ]
  then
    echo "Saving $f to $original ..."
    cp $f $original
  fi
  echo "Patching $f with $f_patch ..."
  if [ -f "$f_patch" ]
  then
    $cmd $f < $f_patch > /dev/null
  else
    echo "Warning: file $f_patch not found!"
  fi
}
```

- Download: https://paulojeronimo.github.io/jboss-domain-mode-scripts/scripts/functions

6.7. scripts/install

```
#!/bin/bash
BASE_DIR=`cd "$(dirname "$0")"; pwd`
source "$BASE_DIR"/common

cd "$BASE_DIR"

./jboss-dependencies-install
./jboss-jdk-install
./jboss-install
./jboss-firewall-configure
```

- Download: https://paulojeronimo.github.io/jboss-domain-mode-scripts/scripts/install
- Dependências:
  - scripts/common
  - scripts/jboss-dependencies-install
  - scripts/jboss-jdk-install
  - scripts/jboss-install
  - scripts/jboss-firewall-configure

6.8. scripts/jboss-add-user

```
#!/bin/bash
BASE_DIR=`cd "$(dirname "$0")"; pwd`
source "$BASE_DIR"/common

echo "Adding JBoss admin user ($JBOSS_ADMIN_USER) ..."
$JBOSS_HOME/bin/add-user.sh -s -u "$JBOSS_ADMIN_USER" -p "$JBOSS_ADMIN_PASSWORD"
```

- Download: https://paulojeronimo.github.io/jboss-domain-mode-scripts/jboss-add-user
- Dependências:
  - scripts/common

6.9. scripts/jboss-backup2master

```
#!/bin/bash
BASE_DIR=`cd "$(dirname "$0")"; pwd`
source "$BASE_DIR"/common

$JBOSS_INSTALL_DIR/$JBOSS_LINK/bin/jboss-cli.sh -c \
--controller=$JBOSS_DOMAIN_MASTER_ADDRESS \
--commands="/host=backup:write-local-domain-controller,reload --host=backup"
```

- Download: https://paulojeronimo.github.io/jboss-domain-mode-scripts/jboss-backup2master
- Dependências:
  - scripts/common

6.10. scripts/jboss-backup2original

```
#!/bin/bash
BASE_DIR=`cd "$(dirname "$0")"; pwd`
source "$BASE_DIR"/common

f=domain/configuration/host-backup.xml
cp $INSTALLERS_DIR/$JBOSS_DIR/$f $JBOSS_INSTALL_DIR/$JBOSS_LINK/$f
chown $JBOSS_USER: $JBOSS_INSTALL_DIR/$JBOSS_LINK/$f
service jboss-as-domain restart
```

- Download: https://paulojeronimo.github.io/jboss-domain-mode-scripts/jboss-backup2original
- Dependências:
  - scripts/common

6.11. scripts/jboss-configure

```
#!/bin/bash
BASE_DIR=`cd "$(dirname "$0")"; pwd`
source "$BASE_DIR"/common

d=$JBOSS_INSTALL_DIR/$JBOSS_DIR
echo "Configuring $d files ..."
cd "$d"

f=bin/domain.conf
case "$JBOSS_TYPE" in
  master) f_patch=$f.master;;
  backup|slave) f_patch=$f.slave;;
esac
patch $f $f_patch

echo "  Configuring JBOSS variables in $f ..."
echo "    JBOSS_BIND_ADDRESS_MANAGEMENT=$JBOSS_BIND_ADDRESS_MANAGEMENT"
echo "    JBOSS_BIND_ADDRESS=$JBOSS_BIND_ADDRESS"
echo "    JBOSS_DOMAIN_MASTER_ADDRESS=$JBOSS_DOMAIN_MASTER_ADDRESS"
[ "$JBOSS_DOMAIN_BACKUP_ADDRESS" ] && \
echo "    JBOSS_DOMAIN_BACKUP_ADDRESS=$JBOSS_DOMAIN_BACKUP_ADDRESS"

sed -i "
  s/JBOSS_BIND_ADDRESS_MANAGEMENT/$JBOSS_BIND_ADDRESS_MANAGEMENT/g ;
  s/JBOSS_BIND_ADDRESS/$JBOSS_BIND_ADDRESS/g ;
  s/JBOSS_DOMAIN_MASTER_ADDRESS/$JBOSS_DOMAIN_MASTER_ADDRESS/g ;
  s/JBOSS_DOMAIN_BACKUP_ADDRESS/$JBOSS_DOMAIN_BACKUP_ADDRESS/g
" $f

if [ "$JBOSS_TYPE" = master ]
then
  f=domain/configuration/domain.xml
  patch $f $f
fi

# tag::host-slave[]
f=domain/configuration/$JBOSS_HOST_CONFIG
f_patch=$f
if [ "$JBOSS_HOST_CONFIG" = "host-slave.xml" ]
then
  [ "$JBOSS_DOMAIN_BACKUP_ADDRESS" ] || f_patch=$f_patch.no-backup
fi
[ "$JBOSS_SERVER_CONFIG" ] && f_patch=$f_patch.$JBOSS_SERVER_CONFIG
patch $f $f_patch
# end::host-slave[]

JBOSS_ADMIN_PASSWORD_BASE64=`echo -n $JBOSS_ADMIN_PASSWORD|base64`
echo "  Configuring JBOSS variables in $f ..."
```

```
echo "    JBOSS_HOST=$JBOSS_HOST"
echo "    JBOSS_ADMIN_USER=$JBOSS_ADMIN_USER"
echo "    JBOSS_ADMIN_PASSWORD_BASE64=$JBOSS_ADMIN_PASSWORD_BASE64"
sed -i "
  s/JBOSS_ADMIN_USER/$JBOSS_ADMIN_USER/g ;
  s/JBOSS_ADMIN_PASSWORD_BASE64/$JBOSS_ADMIN_PASSWORD_BASE64/g ;
  s/JBOSS_HOST/$JBOSS_HOST/g
" $f
```

- Download: https://paulojeronimo.github.io/jboss-domain-mode-scripts/jboss-configure
- Dependências:
  - scripts/common

6.12. scripts/jboss-copy-patches

```
#!/bin/bash
BASE_DIR=`cd "$(dirname "$0")"; pwd`
LINUX_NEEDED=false ROOT_NEEDED=false source "$BASE_DIR"/common

cd "$BASE_DIR/$INSTALLERS_DIR"
echo "Copying $JBOSS_DIR/*.patch (recursively and with parents) to $CONFIGURATIONS_DIR
..."

[ -d "$JBOSS_DIR" ] || { echo "Directory $JBOSS_DIR not found!"; exit 1; }

tar cf - $(find "$JBOSS_DIR" -type f -name '*.patch') | tar xf - -C
"$CONFIGURATIONS_DIR"
```

- Download: https://paulojeronimo.github.io/jboss-domain-mode-scripts/jboss-copy-patches
- Dependências:
  - scripts/common

6.13. scripts/jboss-dependencies-install

```
#!/bin/bash
BASE_DIR=`cd "$(dirname "$0")"; pwd`
source "$BASE_DIR"/common

yum -y install vim rsync zip unzip tree lsof bzip2 telnet patch tmux
```

- Download: https://paulojeronimo.github.io/jboss-domain-mode-scripts/jboss-dependencies-install
- Dependências:
  - scripts/common

6.14. scripts/jboss-extract

```
#!/bin/bash
#
# 1) Extract a fresh copy of JBoss EAP on $INSTALLER_DIR
# 2) Apply JBoss EAP patches
#
# This script can be called outside Linux (ex.: macOS) and does needs to be executed
by root
#
BASE_DIR=`cd "$(dirname "$0")"; pwd`
LINUX_NEEDED=false ROOT_NEEDED=false source "$BASE_DIR"/common

cd "$BASE_DIR/$INSTALLERS_DIR"
echo "Creating base installation ($PWD/$JBOSS_DIR) from $JBOSS_INSTALLER_ZIP ..."

if [ -d "$JBOSS_DIR" ]
then
  echo "Removing existing ..."
  rm -rf "$JBOSS_DIR"
fi

echo "Extracting JBoss from $JBOSS_INSTALLER_ZIP ..."
unzip -q $JBOSS_INSTALLER_ZIP

echo "Patching JBoss ..."
for patch in ${JBOSS_PATCHES[*]}
do
  echo "  Applying patch $patch ..."
  $JBOSS_DIR/bin/jboss-cli.sh --command="patch apply --override-all $patch" >
/dev/null
done

echo "Copying configurations from $CONFIGURATIONS_DIR ..."
rsync -a "$CONFIGURATIONS_DIR/$JBOSS_DIR" .
```

- Download: https://paulojeronimo.github.io/jboss-domain-mode-scripts/scripts/jboss-extract
- Dependências:
  - scripts/common

6.15. scripts/jboss-firewall-configure

```
#!/bin/bash
BASE_DIR=`cd "$(dirname "$0")"; pwd`
source "$BASE_DIR"/common

for port in 9990 9999 8080 8443
do
  firewall-cmd --zone=public --add-port=$port/tcp --permanent
done
firewall-cmd --reload
```

- Download: https://paulojeronimo.github.io/jboss-domain-mode-scripts/scripts/jboss-firewall-configure
- Dependências:
  - scripts/common

6.16. scripts/jboss-fix-permissions

```
#!/bin/bash
BASE_DIR=`cd "$(dirname "$0")"; pwd`
source "$BASE_DIR"/common

echo "Configuring JBoss owner ($JBOSS_USER:$JBOSS_GROUP) ..."
chown -R $JBOSS_USER:$JBOSS_GROUP "$JBOSS_INSTALL_DIR/$JBOSS_DIR"
```

- Download: https://paulojeronimo.github.io/jboss-domain-mode-scripts/scripts/jboss-fix-permissions
- Dependências:
  - scripts/common

6.17. scripts/jboss-install

```
#!/bin/bash
BASE_DIR=`cd "$(dirname "$0")"; pwd`
source "$BASE_DIR"/common

cd "$BASE_DIR"

echo "Installing JBoss Host ($JBOSS_HOST: $JBOSS_TYPE) ..."

./jboss-useradd
./jboss-uninstall

[ -d "$INSTALLERS_DIR/$JBOSS_DIR" ] || ./jboss-extract

echo "Copying $INSTALLERS_DIR/$JBOSS_DIR to $JBOSS_INSTALL_DIR ..."
rsync -a "$INSTALLERS_DIR/$JBOSS_DIR" "$JBOSS_INSTALL_DIR"

echo "Creating link $JBOSS_INSTALL_DIR/$JBOSS_LINK ..."
cd "$JBOSS_INSTALL_DIR"
ln -s $JBOSS_DIR $JBOSS_LINK
cd - &> /dev/null

./jboss-configure
./jboss-add-user
./jboss-fix-permissions
./jboss-service-install
```

- Download: https://paulojeronimo.github.io/jboss-domain-mode-scripts/scripts/jboss-install
- Dependências:
  - scripts/common
  - scripts/jboss-useradd
  - scripts/jboss-uninstall
  - scripts/jboss-configure
  - scripts/jboss-add-user
  - scripts/jboss-fix-permissions
  - scripts/jboss-service-install

6.18. scripts/jboss-jdk-install

```
#!/bin/bash
BASE_DIR=`cd "$(dirname "$0")"; pwd`
source "$BASE_DIR"/common

cd "$BASE_DIR"

if ! which java &> /dev/null
then
  echo "Installing $INSTALLERS_DIR/$JDK_INSTALLER ..."
  rpm -ivh "$INSTALLERS_DIR/$JDK_INSTALLER"
fi
```

- Download: https://paulojeronimo.github.io/jboss-domain-mode-scripts/jboss-jdk-install
- Dependências:
  - scripts/common

6.19. scripts/jboss-service-install

```
#!/bin/bash
BASE_DIR=`cd "$(dirname "$0")"; pwd`
source "$BASE_DIR"/common

cd "$BASE_DIR"

f=$JBOSS_INSTALL_DIR/$JBOSS_DIR/bin/init.d/jboss-as-domain.sh
f2=/etc/init.d/jboss-as-domain
echo "Copying $f to $f2 ..."
[ -r $f2 ] && rm -f $f2
cp "$f" "$f2"

f=/etc/jboss-as/jboss-as.conf
echo "Configuring $f ..."
mkdir -p `dirname "$f"`
cat > "$f" <<EOF
JBOSS_USER=$JBOSS_USER
JBOSS_CONSOLE_LOG=/var/log/jboss-as/console.log
JBOSS_HOME=$JBOSS_HOME
JBOSS_HOST_CONFIG=$JBOSS_HOST_CONFIG
EOF

if [ $JBOSS_TYPE = backup ]
then
  echo "JBOSS_SCRIPT='$JBOSS_INSTALL_DIR/$JBOSS_LINK/bin/domain.sh --backup'" >> $f
fi

if ! chkconfig --list jboss-as-domain &> /dev/null
then
  echo "Adding jboss-as-domain to chkconfig ..."
  chkconfig --add jboss-as-domain
  chkconfig jboss-as-domain on
fi
```

- Download: https://paulojeronimo.github.io/jboss-domain-mode-scripts/scripts/jboss-service-install
- Dependências:
  - scripts/common

6.20. scripts/jboss-start

```
#!/bin/bash
BASE_DIR=`cd "$(dirname "$0")"; pwd`
source "$BASE_DIR"/common

service jboss-as-domain start
```

- Download: https://paulojeronimo.github.io/jboss-domain-mode-scripts/scripts/jboss-start
- Dependências:
  - scripts/common

6.21. scripts/jboss-status

```
#!/bin/bash
BASE_DIR=`cd "$(dirname "$0")"; pwd`
source "$BASE_DIR"/common

service jboss-as-domain status
```

- Download: https://paulojeronimo.github.io/jboss-domain-mode-scripts/scripts/jboss-status
- Dependências:
  - scripts/common

6.22. scripts/jboss-stop

```
#!/bin/bash
BASE_DIR=`cd "$(dirname "$0")"; pwd`
source "$BASE_DIR"/common

service jboss-as-domain stop
```

- Download: https://paulojeronimo.github.io/jboss-domain-mode-scripts/scripts/jboss-stop
- Dependências:
  - scripts/common

6.23. scripts/jboss-tailf

```
#!/bin/bash
BASE_DIR=`cd "$(dirname "$0")"; pwd`
source "$BASE_DIR"/common

tailf "$JBOSS_CONSOLE_LOG"
```

- Download: https://paulojeronimo.github.io/jboss-domain-mode-scripts/scripts/jboss-tailf
- Dependências:
  - scripts/common

6.24. scripts/jboss-uninstall

```
#!/bin/bash
BASE_DIR=`cd "$(dirname "$0")"; pwd`
source "$BASE_DIR"/common

cd "$BASE_DIR"

./jboss-service-uninstall

d=$JBOSS_INSTALL_DIR/$JBOSS_DIR
if [ -d "$d" ]
then
  echo "Removing $d ..."
  rm -rf "$d"
fi

f=$JBOSS_INSTALL_DIR/$JBOSS_LINK
if [ -L "$f" ]
then
  echo "Removing link $f ..."
  rm "$f"
fi
```

- Download: https://paulojeronimo.github.io/jboss-domain-mode-scripts/scripts/jboss-uninstall
- Dependências:
  - scripts/common

6.25. scripts/jboss-useradd

```
#!/bin/bash
BASE_DIR=`cd "$(dirname "$0")"; pwd`
source "$BASE_DIR"/common

cd "$BASE_DIR"

grep -q $JBOSS_GROUP /etc/group || ./jboss-groupadd

if ! grep -q $JBOSS_USER /etc/passwd
then
  useradd -r $JBOSS_USER -g $JBOSS_GROUP && echo "JBOSS_USER ($JBOSS_USER) created!"
  passwd --stdin $JBOSS_USER <<< $JBOSS_USER_PASSWORD
else
  echo "JBOSS_USER ($JBOSS_USER) already created!"
fi
```

- Download: https://paulojeronimo.github.io/jboss-domain-mode-scripts/scripts/jboss-useradd
- Dependências:
  - scripts/common

6.26. scripts/jboss-userdel

```
#!/bin/bash
BASE_DIR=`cd "$(dirname "$0")"; pwd`
source "$BASE_DIR"/common

cd "$BASE_DIR"

if grep -q $JBOSS_USER /etc/passwd
then
  userdel $JBOSS_USER && echo "JBOSS_USER ($JBOSS_USER) removed!"
else
  echo "JBOSS_USER ($JBOSS_USER) already removed!"
fi
```

- Download: https://paulojeronimo.github.io/jboss-domain-mode-scripts/scripts/jboss-userdel
- Dependências:
  - scripts/common

7. Soluções de exercícios

7.1. Demo 2, exercício 1

Possível solução para "Exercício 1: substituição do instalador, reinstalação e inicialização do DC (dc1)":

```
alias dc1ssh='sshpass -p vagrant ssh -p 2222 root@$SERVER' ①

②
export SERVER=localhost

③
SSH_PORT=2222 ./jboss-installer.install

④
dc1ssh $scripts/jboss-install ⑤
dc1ssh $scripts/jboss-start
```

① Diferenças desse `alias` para o apresentado na "Demo 1": 1) Estamos utilizando o `sshpass` para não sermos solicitados a entrar a senha do usuário `root`. 2) Estamos utilizando a variável `SERVER`.

② Criamos e exportamos a variável `SERVER` para não precisarmos mais ficar digitando-a na frente dos scripts.

③ Não precisamos mais passar `SERVER` como parâmetro.

④ Para ser utilizada a partir desse ponto, a variável `scripts` já precisaria estar definida.

⑤ Observe que não precisamos fazer a instalação completa (via scripts/install) pois o `dc1` já estava instalado. Fizemos isso na "Demo 1".

7.2. Demo 2, exercício 2

Solução 1 para "Exercício 2: reinstalação do hc1 e instalação do hc2" (execute-a num shell Bash escrevendo-a linha a linha):

```
export SSHPASS=vagrant ①

alias hc1ssh='sshpass -e ssh -p 2200 root@$SERVER'
alias hc2ssh='sshpass -e ssh -p 2201 root@$SERVER'

for p in 220{0,1}; do SSH_PORT=$p ./jboss-installer.install; done ②

hc1ssh $scripts/jboss-install
hc1ssh $scripts/jboss-start
hc1ssh $scripts/jboss-tailf &> /tmp/hc1ssh.log & ③

hc2ssh $scripts/install ④
hc2ssh $scripts/jboss-start
hc2ssh $scripts/jboss-tailf &> /tmp/hc2ssh.log &
```

① A variável `SSHPASS` pode ser utilizada no lugar de passarmos uma senha via linha de comando. Ela também está sendo utilizada pelo jboss-installer.install.

② Esse `loop` é executado de forma serializada (performance ruim).

③ Esse comando redirecionará os últimos logs de `hc1` para um arquivo temporário.

④ No `hc2`, de forma contrária ao `hc1`, precisamos fazer a instalação completa (é a sua primeira instalação).