DIMENSA

PROCEDIMENTO DE INSTALAÇÃO

WILDFLY

SUMÁRIO

HISTÓRICO DE REVISÃO

Data	Versão	Autor	Descrição
20/07/2018	1.0	Time de Arquitetura	Criação do documento
24/07/2018	1.1	Time de Arquitetura	Revisão do documento
23/08/2018	1.2	Ivan Pereira	Ajuste versão Wildfly 11
15/03/2024	2.0	Ivan Pereira	Alteração de layout Atualização dos procedimentos de instalação

1. Objetivo

Este documento tem como objetivo descrever o processo de configuração do servidor de aplicação Wildfly, responsável pela execução dos serviços e portais, permitindo técnicos e administradores de sistemas executarem os procedimentos de implantação dos sistemas DIMENSA. É requerido que os executores estejam habituados com rotinas de instalação e configuração de Sistemas Operacionais, Banco de Dados e Servidores de Aplicação, bem como as tarefas de implantação de aplicações.

Será necessário um DBA ou um profissional técnico com conhecimento e habilidade para executar as tarefas relacionadas ao Banco de Dados.

2. Arquitetura

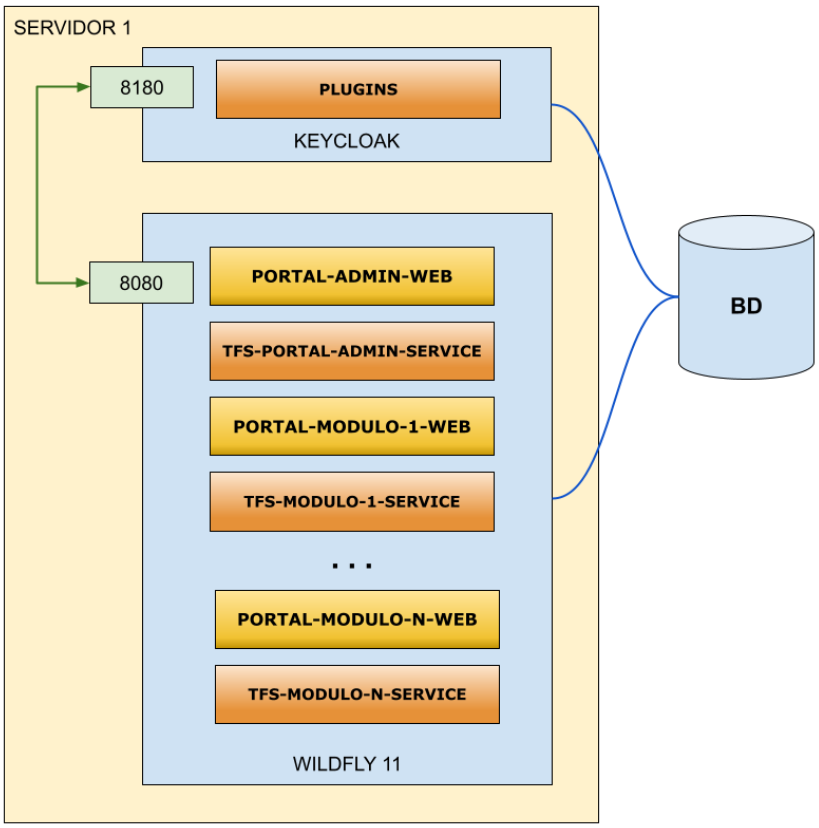
De acordo com cada caso, a arquitetura a ser utilizada por variar, por isto sugerimos três cenários para implantação dos servidores de aplicação e autenticação no ambiente. Entender e definir esta arquitetura inicial influencia nas configurações a serem realizadas nos passos posteriores deste documento.

É importante ressaltar que as configurações definidas nessa arquitetura são mínimas e necessárias para o funcionamento do ambiente em implantação.

2.1. Cenário 1 – Modelo Servidor Único

Neste modelo de arquitetura, o servidor de aplicação e autenticação são executados no mesmo servidor. Recomendado para ambientes onde são executados poucos módulos, com baixa utilização e poucos usuários simultâneos. Os recursos do servidor deverão ser dimensionados de acordo com o número de módulos implantados.

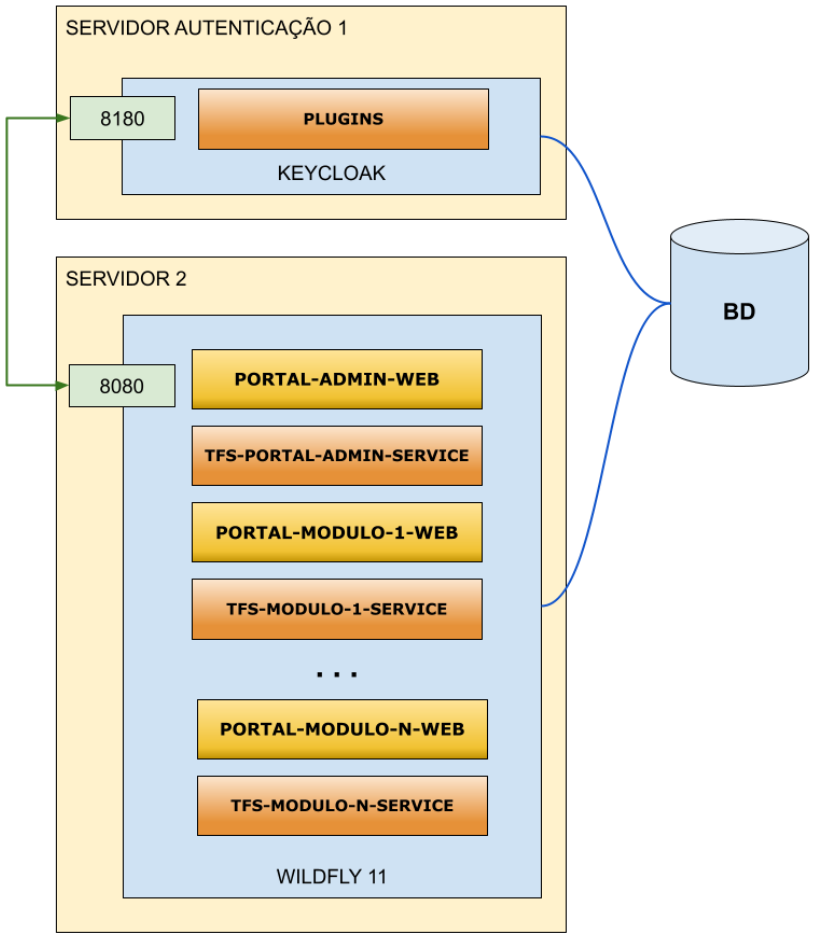
O diagrama do modelo de Servidor único está ilustrado abaixo:



2.2. Cenário 2 – Modelo Segregado

Neste modelo de arquitetura, os servidores de aplicação ficam separados do servidor de autenticação, podendo ter 1 ou mais servidores de aplicação utilizando o mesmo servidor de autenticação. Recomendado para ambientes onde o número de módulos e usuários são um pouco maiores, mas que não necessitem de distribuição de carga nem alta disponibilidade.

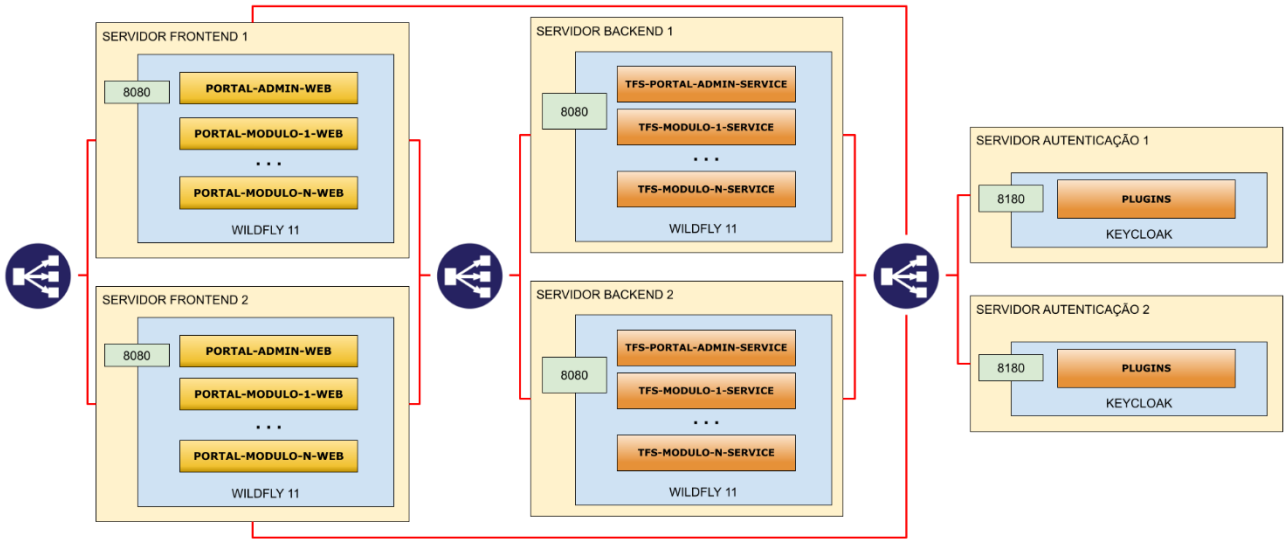
Este modelo segregado está ilustrado no diagrama abaixo:



2.3. Cenário 3 – Modelo Alta Disponibilidade

Neste modelo de arquitetura, os servidores de aplicação e os servidores de autenticação operam no modo cluster tendo suas conexões distribuídas por um balanceador de carga, podendo utilizar dois ou mais servidores de autenticação. Esta arquitetura é recomendada para ambientes onde exige-se um número maior de acessos aos portais, serviços da aplicação e de autenticação.

Uma sugestão do modelo de alta disponibilidade está ilustrada no diagrama abaixo:



3. Pré-Requisitos

Sistema Operacional

Este produto é compatível com a maioria das distribuições do Linux, contudo pode haver pequenas diferenças de comportamento nos sistemas que ainda não foram homologados pela Dimensa. Por isso, recomendamos que sejam utilizados os seguintes sistemas operacionais:

- RedHat Enterprise Linux 8 ou superior
- Oracle Enterprise Linux 8 ou superior
- CentOS 7 ou superior
- OpenSuse Leap 15 ou superior
- Ubuntu Server 20.04 LTS ou superior
- Debian 12 ou superior

A instalação do sistema operacional, bem como sua administração são de responsabilidade do cliente e os passos descritos neste manual somente contemplam os ajustes necessários para a execução das aplicações que serão instaladas.

Java 8

É necessário realizar previamente a instalação da JDK 1.8.0_202 (gratuita) ou, caso possua licença da Oracle, uma versão superior. A utilização da OpenJDK 1.8.0_362 ou superior também é uma possibilidade.

Usuário de Sistema

Para executar o Wildfly recomenda-se a criação de um usuário exclusivo ou ainda utilizar um usuário de serviços já existente. Por questões de segurança, o Wildfly não deverá ser executado com o usuário root ou outro usuário com poderes equivalentes. Para exemplificar, vamos utilizar o usuário **wildfly**.

4. Procedimento de Instalação

4.1. Banco de Dados

O sistema de autenticação utiliza o database ou schema SEGURANCA, podendo variar o nome de acordo com o ambiente (ex. TOTVS_SEGURANCA, DIMENSA_SEGURANCA, etc). O Keycloak efetua a criação das suas tabelas automaticamente, porém existem outras tabelas utilizadas por outros módulos. Os scripts destas outras tabelas são disponibilizados durante a etapa de implantação.

4.2. Servidor de Aplicação Wildfly

O Wildfly é um servidor de aplicação responsável por executar artefatos criados em Java e será fornecido o arquivo compactado contendo as configurações padrões. Este arquivo deverá ser instalado no diretório **/opt/app/wildfly-11.0.0.Final** e o mesmo deverá pertencer ao usuário **wildfly** ou outro usuário de serviço estipulado na seção de pré-requisitos. É possível também ajustar a ACL do diretório para o usuário, desde que o mesmo tenha permissão de leitura, escrita e execução. Deve-se verificar se todos os arquivos com extensão .sh no diretório **/opt/app/wildfly-11.0.0.Final/bin** estão com permissão de execução. Dependendo da arquitetura escolhida, é necessário ter várias instâncias do Wildfly rodando em portas diferentes e utilizar um proxy reverso (Nginx) para centralizar em um único endereço e porta.

4.3. Arquivo standalone.conf

O arquivo **standalone.conf** está localizado no diretório **/opt/app/wildfly-11.0.0.Final/bin** e é onde se define as configurações do **JAVA_HOME**, **JAVA_OPTS** e **SERVER_OPTS**. Caso a JDK ou a OpenJDK esteja no **PATH**, a variável **JAVA_HOME** poderá ser comentada ou retirada do arquivo, caso contrário é necessário informar o caminho da mesma. Os parâmetros de memória na variável **JAVA_OPTS** poderá ser ajustado de acordo com o tamanho do ambiente, a carga no servidor de aplicação e a quantidade de módulos que serão implantados. O valor inicial recomendado é de no mínimo 1Gb caso haja implantação apenas de portais e de no mínimo de 2Gb para implantações de serviços.

Abaixo valores sugeridos para cada uma dessas variáveis:

```
JAVA_HOME=”/opt/jdk1.8.0_202”
JAVA_OPTS=”-Xms2G -Xmx2G”
JAVA_OPTS=”$JAVA_OPTS -Djava.net.preferIPv4Stack=true -Duser.language=pt -Duser.country=BR”
```

4.4. Arquivo standalone.xml

Neste arquivo são definidas as configurações de rotação de logs, configuração de datasource, portas, entre outros e o mesmo está localizado no diretório **/opt/app/wildfly-11.0.0.Final/standalone/configuration**.

4.4.1. Configuração da Rotação de Log

Dentro da tag `<subsystem xmlns="urn:jboss:domain:logging:3.0">` é possível efetuar a configuração de rotação de logs, permitindo parametrizar o tipo, padrão, tamanho máximo, entre outros. Normalmente configura-se a rotação por tamanho de arquivo, conforme exemplo abaixo, onde o tamanho do arquivo está para 50Mb e serão mantidos os últimos 5 arquivos.

```
<size-rotating-file-handler name="FILE">
	<formatter>
		<pattern-formatter pattern="%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%E%n"/>
	</formatter>
	<file relative-to="jboss.server.log.dir" path="server.log"/>
	<rotate-size value="50m"/>
	<max-backup-index value="5"/>
	<append value="true"/>
</size-rotating-file-handler>
```

Para utilização do tipo de rotação de logs de periodicidade diária é necessário trocar toda a tag `size-rotating-file-handler` pelas tags abaixo. Neste exemplo, está configurado para criar um arquivo por dia e quando o mesmo atingir 200Mb, é gerado um novo arquivo com a mesma data anexando uma numeração no final do arquivo, limitando-se a 10 arquivos por dia no máximo.

```
<periodic-size-rotating-file-handler name="FILE" autoflush="true">
	<encoding value="UTF-8"/>
	<formatter>
		<named-formatter name="PATTERN"/>
	</formatter>
	<file relative-to="jboss.server.log.dir" path="server.log"/>
	<rotate-size value="200M"/>
	<max-backup-index value="10"/>
	<suffix value=".yyyy-MM-dd"/>
	<append value="true"/>
</periodic-size-rotating-file-handler>
```

4.4.2. Configuração do Datasource

Os procedimentos abaixo apresentam as informações de configuração do Datasource, local onde é informado qual servidor de banco de dados será usado, bem como usuário e senha de conexão.

As informações abaixo correspondem as configurações aplicadas no servidor Wildfly de acordo com o tipo de banco de dados utilizado.

Oracle

Para configurar o datasource para o banco de dados Oracle, é necessário substituir as seguintes informações no exemplo abaixo.

HOST – Endereço IP ou hostname do servidor Oracle

PORTA – Porta do servidor, normalmente 1521

SERVICE-NAME – Service Name do banco de dados

USUARIO_BD – Deve-se utilizar o owner APP ou equivalente

SENHA_BD – Senha do owner.

```
<datasource jndi-name="java:jboss/datasources/KeycloakDS" pool-name="KeycloakDS"
enabled="true" use-java-context="true">
	<connection-url>jdbc:oracle:thin:@//HOST:PORTA/SERVICE-NAME<connection-url>
	<driver>oracle</driver>
	<pool>
		<min-pool-size>10</min-pool-size>
		<max-pool-size>60</max-pool-size>
		<prefill>false</prefill>
		<flush-strategy>FailingConnectionOnly</flush-strategy>
	</pool>
	<security>
		<user-name>USUARIO_BD</user-name>
		<password>SENHA_BD</password>
	</security>
	<validation>
		<check-valid-connection-sql>SELECT 1 FROM REALM</check-valid-connection-sql>
		<validate-on-match>true</validate-on-match>
		<background-validation>true</background-validation>
		<background-validation-millis>1000</background-validation-millis>
		<use-fast-fail>true</use-fast-fail>
	</validation>
	<timeout>
		<set-tx-query-timeout>false</set-tx-query-timeout>
		<blocking-timeout-millis>0</blocking-timeout-millis>
		<idle-timeout-minutes>0</idle-timeout-minutes>
		<query-timeout>0</query-timeout>
		<use-try-lock>0</use-try-lock>
		<allocation-retry>0</allocation-retry>
		<allocation-retry-wait-millis>0</allocation-retry-wait-millis>
	</timeout>
</datasource>
```

MS SQL Server

Para configurar o datasource para o banco de dados MS SQL Server, é necessário substituir as seguintes informações no exemplo abaixo.

HOST – Endereço IP ou hostname do servidor

NOME_BD – Nome do Banco de Dados

NOME_INSTANCIA – Nome da instância

USUARIO_BD – Usuário com permissão para acessar o database APP ou equivalente

SENHA_BD – Senha do usuário.

```
<datasource jndi-name="java:jboss/datasources/KeycloakDS" pool-name="KeycloakDS"
enabled="true" use-java-context="true">
	<connection-
url>jdbc:sqlserver://HOST;DatabaseName=NOME_BD;instanceName=NOME_INSTANCIA;sendStringParam
etersAsUnicode=false<connection-url>
	<driver>sqlserver</driver>
		<pool>
		<min-pool-size>10</min-pool-size>
		<max-pool-size>60</max-pool-size>
		<prefill>false</prefill>
		<flush-strategy>FailingConnectionOnly</flush-strategy>
	</pool>
	<security>
		<user-name>USUARIO_BD</user-name>
		<password>SENHA_BD</password>
	</security>
	<validation>
		<check-valid-connection-sql>SELECT 1 FROM REALM</check-valid-connection-sql>
		<validate-on-match>true</validate-on-match>
		<background-validation>true</background-validation>
		<background-validation-millis>1000</background-validation-millis>
		<use-fast-fail>true</use-fast-fail>
	</validation>
	<timeout>
		<set-tx-query-timeout>false</set-tx-query-timeout>
		<blocking-timeout-millis>0</blocking-timeout-millis>
		<idle-timeout-minutes>0</idle-timeout-minutes>
		<query-timeout>0</query-timeout>
		<use-try-lock>0</use-try-lock>
		<allocation-retry>0</allocation-retry>
		<allocation-retry-wait-millis>0</allocation-retry-wait-millis>
	</timeout>
</datasource>
```

4.4.3. Offset de Portas

O Wildfly por padrão utiliza a porta 8080, porém há várias situações onde é necessário executar o em portas diferentes da convencional, tais como a convivência com outros servidores de aplicação, muito comum no caso da segregação de serviços por servidores.

O offset de portas é um valor definido via variável de parâmetros ou diretamente no arquivo standalone.xml, onde na tag socket-binding-group, há o parâmetro port-offset. O número informado ali, será somado ao valor de referência registrado na variável jboss.http.port (8080).

```
<socket-binding-group name="standard-sockets" default-interface="public" port-offset="${jboss.socket.binding.port-offset:0}">
    <socket-binding name="management-http" interface="management" port="${jboss.management.http.port:9990}"/>
    <socket-binding name="management-https" interface="management" port="${jboss.management.https.port:9993}"/>
    <socket-binding name="ajp" port="${jboss.ajp.port:8009}"/>
    <socket-binding name="http" port="${jboss.http.port:8080}"/>
    <socket-binding name="https" port="${jboss.https.port:8443}"/>
    <socket-binding name="txn-recovery-environment" port="4712"/>
    <socket-binding name="txn-status-manager" port="4713"/>
    <outbound-socket-binding name="mail-smtp">
        <remote-destination host="localhost" port="25"/>
    </outbound-socket-binding>
</socket-binding-group>
```

4.5. Serviço de Inicialização

Para o Wildfly executar como serviço utilizamos o gerenciador de serviços **systemd** do Linux. Os passos abaixo são para criar e habilitar o serviço para iniciar o servidor de autenticação.

Inicialmente deve-se criar o arquivo **wildfly.service** no diretório **/etc/systemd/system** e para isto é necessário estar logado com o usuário **root** ou algum usuário com poder equivalente a super usuário. O conteúdo do arquivo deve ser o seguinte:

```
[Unit]
Description=DIMENSA - Wildfly 11.0.0-Final

[Service]
Type=simple
User=wildfly
RemainAfterExit=yes
ExecStart=/opt/app/wildfly-11.0.0.Final/bin/standalone.sh
Restart=on-failure
RestartSec=120s

[Install]
WantedBy=multi-user.target
```

Após salvar o arquivo, é necessário digitar o seguinte comando para carregar o novo serviço adicionado:

```
systemctl daemon-reload
```

Para habilitar a execução automática do serviço do Wildfly durante a inicialização do servidor, é necessário digitar o seguinte comando:

```
systemctl enable wildfly
```

Para iniciar o serviço do Wildfly, é necessário digitar o seguinte comando:

```
systemctl start wildfly
```

5. Implantação e Atualização de Artefatos

Para implantar ou atualizar as aplicações, utilizamos o hot-deploy do Wildfly, que consiste em copiar o arquivo para o diretório **/opt/app/wildfly-11.0.0.Final/standalone/deployments/** e aguardar o processo de implantação. Cada arquivo com extensão WAR e EAR possui um arquivo correspondente com a extensão **deployed**, **dodeploy**, **isdeploying, undeployed** ou **failed**. Quando todos os artefatos tiverem um arquivo com a extensão **deployed**, a implantação estará concluída. Caso algum fique com a extensão **failed**, é necessário verificar o log do servidor de aplicação para diagnosticar o motivo da falha.