

Websphere Liberty - Introdução ao Modo Dev

"App Modernization não é só sobre runtimes". É Developer Experience ...



Última atualização: Setembro de 2024

Duração: 30 minutes

1. Objetivos

Neste exercício, você aprenderá como os desenvolvedores podem usar o modo Liberty no modo "dev" para obter desenvolvimento iterativo eficiente, teste, ciclo de depuração ao desenvolver aplicações / microsserviços baseados em Java.

No final deste laboratório você deve ser capaz de:

- Use o modo Liberty dev (independente) sem uma IDE
- Experimentar o recarregamento quente de código de aplicação e alterações de configuração usando o modo dev
- Trabalhar com o modo Liberty dev em contêineres
- Executar testes de unidade / integração integrados a partir do modo Liberty dev

2. Introdução-Liberty e Modo "dev"

Open Liberty é um servidor de aplicativos projetado para a nuvem. É pequeno, leve e projetado com desenvolvimento moderno de aplicações nativas.

Open Liberty suporta as APIs completas do MicroProfile e Jakarta EE, significando que você pode usar apenas os recursos que você precisa, mantendo o servidor leve, o que é ótimo

para os microserviços. Ele também implementa a cada grande plataforma de nuvem, incluindo Docker, Kubernetes e Cloud Foundry.


Maven é uma ferramenta de construção de automação que fornece uma maneira eficiente de desenvolver aplicações Java. Usando o Maven, você construirá um microserviço simples, chamado sistema, que coleta propriedades básicas do sistema do seu laptop e as exibe em um terminal que você pode acessar em seu navegador web.

O modo de desenvolvimento aberto Liberty, ou modo dev, permite desenvolver aplicativos com qualquer editor de texto ou IDE, fornecendo hot reload e implementação, sob teste de demanda e suporte de depurador. O open Liberty Dev Mode é ativado através de projetos Maven e Gradle.

Seu código é compilado e implementado automaticamente para o seu servidor em execução, facilitando a iteração em suas alterações.

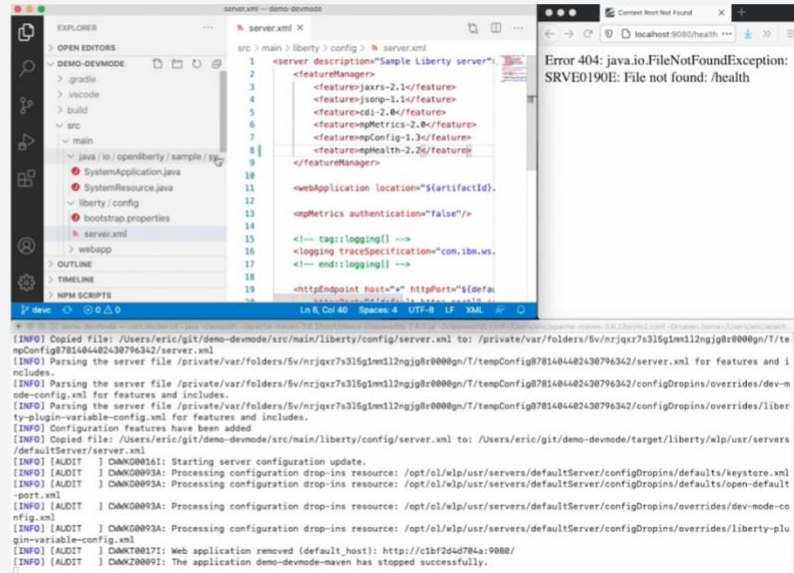
Você pode executar testes sob demanda ou até mesmo automaticamente para que você possa obter feedback imediato sobre suas alterações. Você também pode anexar um depurador a qualquer momento para depurar o seu aplicativo em execução.

Dev mode in action



- Boosts developer productivity
- Immediate feedback for code and config changes
- No re-build necessary
- Hot deployment, testing and debugging
- Including in Containers

```
mvn liberty:dev mvn liberty:dev
gradle libertyDev gradle libertyDev
```



The screenshot shows an IDE with the `server.xml` file open. The file contains configuration for a Liberty server, including features like `jaxrs-2.1`, `jsonp-1.1`, `cdi-2.0`, `metrics-2.0`, `config-1.3`, and `health-2.2`. The `webApplication` element is configured with `location="/"` and `authentication="false"`. The `logging` element is configured with `traceSpecification="com.ibm.ws.*:*=DEBUG"`. The `httpEndpoint` element is configured with `host="*"` and `httpPort="8080"`. The terminal window shows logs for the Liberty server, including the message "The application demo-devmode-maven has stopped successfully."

Neste laboratório, você aprenderá como executar e atualizar um microserviço REST simples em um servidor Open Liberty usando o modo desenvolvedor (modo dev). Você usará o Maven em todo o guia para construir e executar o microserviço bem como para interagir com a instância do servidor em execução.

2.1 Construindo e executando o aplicativo usando Maven e o liberty-maven-plugin

O aplicativo de amostra usado neste laboratório está configurado para ser construído com Maven. Todo projeto configurado pelo Maven contém um arquivo pom.xml, que define a configuração do projeto, dependências, plug-ins e etc.

O seu arquivo pom.xml está localizado no diretório raiz do projeto e está configurado para incluir o liberty-maven-plugin, que permite instalar aplicativos em Open Liberty e gerenciar as instâncias do servidor.

Para começar, navegue até o diretório do projeto. Construa o microserviço "**system**" que é fornecido e implemente-o para o Open Liberty executando o Maven **liberty: run**:

1. Use o Ícone de **Activities** para alternar para a barra de ferramentas, em seguida, clique no ícone Terminal para abrir uma janela do Terminal.



2. Clone o repo GitHub que inclui artefatos necessários para este laboratório

```
mkdir -p /home/techzone/Student/labs

git clone https://github.com/openliberty/guide-getting-started.git /home/techzone/Student/labs/devmode

cd /home/techzone/Student/labs/devmode
```

Uma vez concluído, o repo de artefatos de laboratório local é clonado no seguinte diretório na VM de desktop.

/home/techzone/Student/labs/devmode

3. Para começar, navegue até o diretório inicial. Construa o microserviço "**system**" que é fornecido e implemente-o para o Open Liberty executando o Maven **liberty: run**:
 - a. Abra uma janela de terminal e mude para o diretório **/home/techzone/Student/labs/devmode/start**

```
cd /home/techzone/Student/labs/devmode/start
```

- b. Executar o `liberty mvn`: execute o comando para iniciar o servidor Liberty

```
mvn liberty:run
```

O comando `mvn` inicia um build de Maven, e o diretório de destino é criado para armazenar todos os arquivos relacionados à construção.

O argumento `liberty: run` especifica o objetivo de execução do Open Liberty, que inicia uma instância do servidor Open Liberty em primeiro plano.

Como parte desta fase, um tempo de execução do servidor Open Liberty é baixado e instalado no diretório `target/liberty/wlp`, uma instância do servidor é criada e configurada no diretório `target/liberty/wlp/usr/servers/defaultServer`, e o aplicativo é instalado naquele servidor.

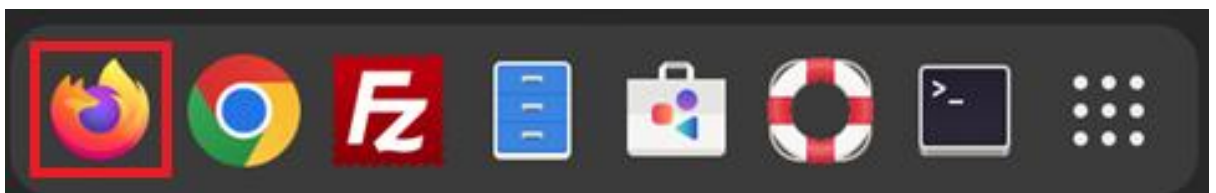
Quando o servidor começa a iniciar, várias mensagens são exibidas em sua sessão de linha de comandos. Aguarde a seguinte mensagem, que indica que a inicialização do servidor está completa:

The server defaultServer is ready to run a smarter planet.

```
[INFO] [AUDIT ] CWPKI0820A: The default keystore has been created using the 'keystore_password' environment variable.
[INFO] [AUDIT ] CWWKS4104A: LTPA keys created in 2.167 seconds. LTPA key file: /home/techzone/Student/labs/devmode/start/target/liberty/wlp/usr/servers/defaultServer/resources/security/ltpa.keys
[INFO] [AUDIT ] CWPKI0803A: SSL certificate created in 3.258 seconds. SSL key file: /home/techzone/Student/labs/devmode/start/target/liberty/wlp/usr/servers/defaultServer/resources/security/key.p12
[INFO] [AUDIT ] CWWKT0016I: Web application available (default_host): http://rhel9-base.gym.lan:9080/ibm/api/
[INFO] [AUDIT ] CWWKT0016I: Web application available (default_host): http://rhel9-base.gym.lan:9080/metrics/
[INFO] [AUDIT ] CWWKT0016I: Web application available (default_host): http://rhel9-base.gym.lan:9080/
[INFO] [AUDIT ] CWWKZ0001I: Application guide-getting-started started in 3.183 seconds.
[INFO] [AUDIT ] CWWKF0012I: The server installed the following features: [cdi-4.0, distributedMap-1.0, jndi-1.0, json-1.0, jsonb-3.0, jsonp-2.1, monitor-1.0, mpConfig-3.1, mpMetrics-5.1, restfulWS-3.1, restfulWSClient-3.1, ssl-1.0, transportSecurity-1.0].
[INFO] [AUDIT ] CWWKF0011I: The defaultServer server is ready to run a smarter planet. The defaultServer server started in 14.530 seconds.
```

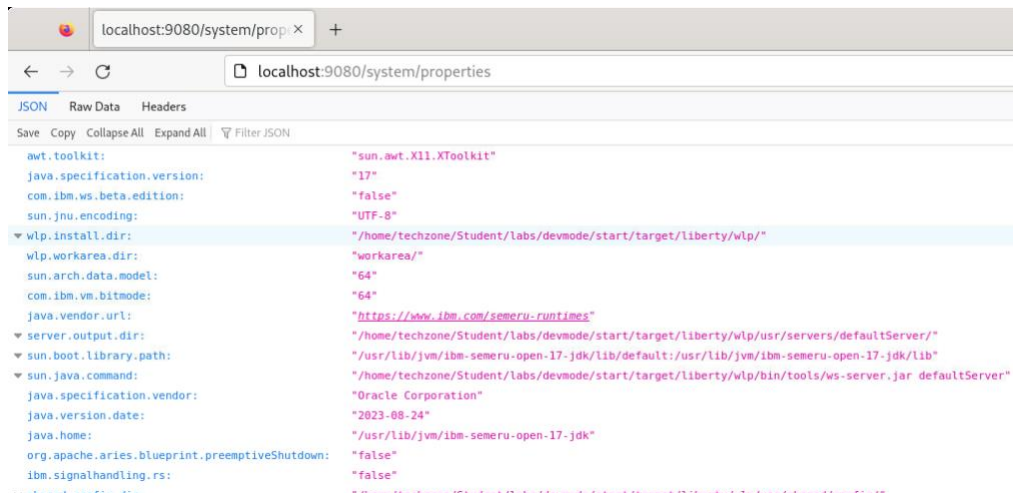
4. Acesse o microserviço " **system** " que foi implementado no servidor Liberty.

- a. Use o ícone de **Activities** para alternar para a barra de ferramentas, em seguida, clique no ícone do Firefox para abrir uma janela do navegador Firefox.



Acesse a URL abaixo. O microserviço lista várias propriedades do sistema da sua JVM.

```
http://localhost:9080/system/properties
```



5. Pare o servidor Liberty, pressionando o CTRL + C na sessão da linha de comandos onde você executou o servidor.
6. Iniciar e Parar o servidor Liberty em segundo plano

Embora você possa iniciar e parar o servidor em primeiro plano usando o Maven **liberty: run**, você também pode iniciar e parar o servidor em segundo plano com o Maven **liberty: start** e **liberty: stop**:

```
mvn liberty:start
mvn liberty:stop
```

7. Visualize o arquivo **pom.xml** para ver o liberty-maven-plugin que foi usado nas etapas anteriores.

- a. A partir de uma janela do Terminal, navegue até o seguinte diretório

```
cd /home/techzone/Student/labs/devmode/start
```

- b. Visualizar o plugin relevante no arquivo **pom.xml**. As opções -A e -B no comando grep exibem o número especificado de linhas antes e depois do local da sequência de texto de pesquisa.

```
cat pom.xml | grep -B 4 -A 2 liberty-maven-plugin
```

```
[techzone@rhel9-base start]$ cat pom.xml | grep -B 4 -A 2 liberty-maven-plugin
    <plugins>
      <!-- Enable liberty-maven plugin -->
      <plugin>
        <groupId>io.openliberty.tools</groupId>
        <artifactId>liberty-maven-plugin</artifactId>
        <version>3.10.3</version>
      </plugin>
```

2.2 Como atualizar o aplicativo sem reiniciar o servidor

O plug-in Open Liberty Maven inclui um objetivo de dev que atende a quaisquer alterações no projeto, incluindo o código-fonte do aplicativo ou alterações de configuração. O servidor Open Liberty recarrega automaticamente o aplicativo e a configuração sem recomeçar. Esse objetivo permite um turnarounds mais rápido e uma melhor experiência de desenvolvedor.

1. Iniciar o servidor Liberty no modo "dev"

- a. A partir de uma janela do Terminal, navegue até o seguinte diretório

```
cd /home/techzone/Student/labs/devmode/start
```

- b. Assegure que o servidor Liberty esteja PARADO!

```
mvn liberty:stop
```

- c. Iniciar Liberty no modo dev

```
mvn liberty:dev
```

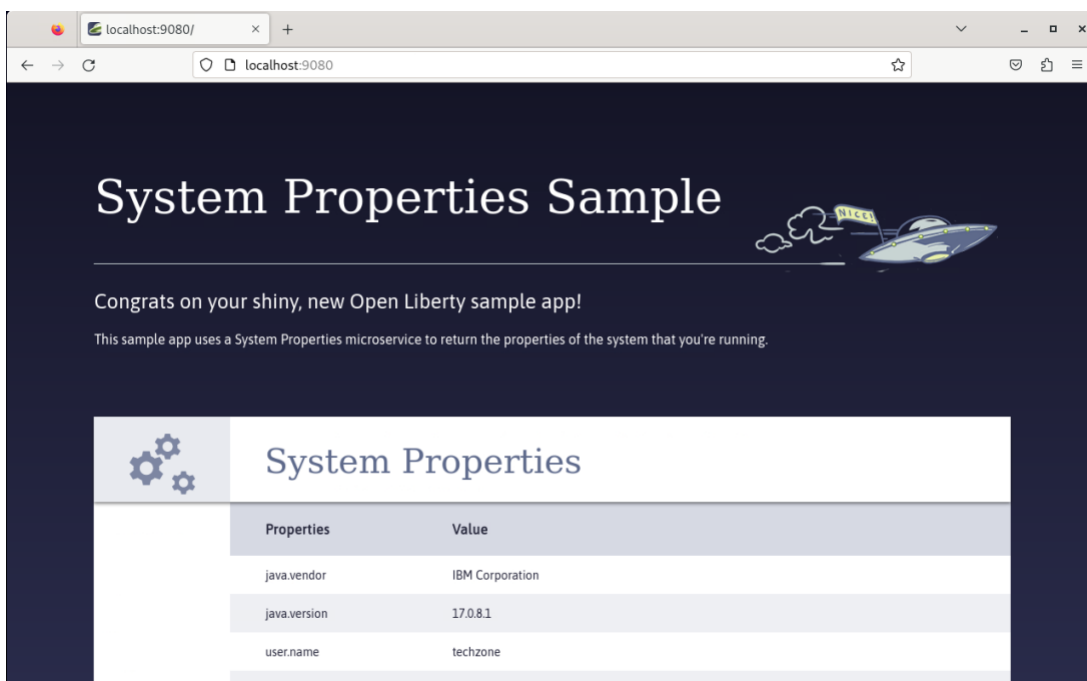
Nota: O Liberty agora é iniciado no modo dev.


```
techzone@rhel9-base:~/Student/labs/devmode/start — /usr/lib/jvm/ibm-semeru-open-17-jdk/bin/java -classpath /home/te...
[INFO] [AUDIT ] CWMKZ0001I: Application guide-getting-started started in 2.396 seconds.
[INFO] [AUDIT ] CWMKF0012I: The server installed the following features: [cdi-4.0, distributedMap-1.0, jndi-1.0, json-1.0, jsonb-3.0, jsonp-2.1, monitor-1.0, mpConfig-3.1, mpMetrics-5.1, restfulWS-3.1, restfulWSClient-3.1, ssl-1.0, transportSecurity-1.0].
[INFO] [AUDIT ] CWMKF0011I: The defaultServer server is ready to run a smarter planet. The defaultServer server started in 6.886 seconds.
[INFO] CWMKM2015I: Match number: 1 is [7/29/24, 8:18:41:435 EDT] 0000002f com.ibm.ws.kernel.feature.internal.FeatureManager A CWMK
F0011I: The defaultServer server is ready to run a smarter planet. The defaultServer server started in 6.886 seconds..
[INFO] *****
[INFO] * Liberty is running in dev mode.
[INFO] * Automatic generation of features: [ Off ]
[INFO] * h - see the help menu for available actions, type 'h' and press Enter.
[INFO] * q - stop the server and quit dev mode, press Ctrl-C or type 'q' and press Enter.
[INFO] *
[INFO] * Liberty server port information:
[INFO] * Liberty server HTTP port: [ 9080 ]
[INFO] * Liberty server HTTPS port: [ 9443 ]
[INFO] * Liberty debug port: [ 7777 ]
[INFO] *****
[INFO] Source compilation was successful.
[INFO] Tests compilation was successful.
[INFO] [AUDIT ] CWMKT0017I: Web application removed (default_host): http://rhel9-base.gym.lan:9080/
[INFO] [AUDIT ] CWMKZ0009I: The application guide-getting-started has stopped successfully.
[INFO] [AUDIT ] CWMKT0016I: Web application available (default_host): http://rhel9-base.gym.lan:9080/
[INFO] [AUDIT ] CWMKZ0003I: The application guide-getting-started updated in 0.545 seconds.
```

O modo dev capta automaticamente as alterações que você faz em seu aplicativo e permite executar testes pressionando a tecla enter / return na sessão da linha de comandos ativa. Quando você estiver trabalhando em seu aplicativo, em vez de redireciona comandos Maven, pressione a tecla enter / return para verificar sua mudança, que executa seus testes.

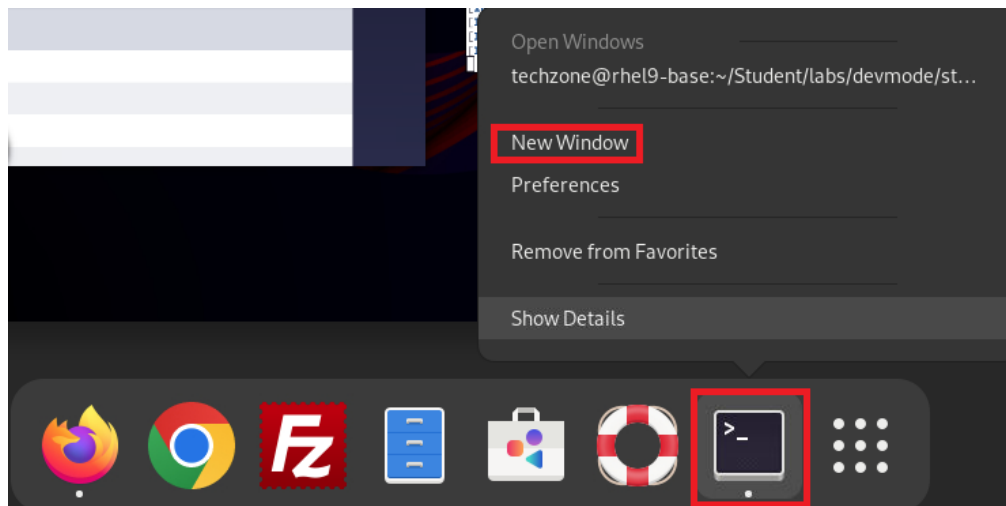
2. Acesse o microserviço "system" que foi implementado no servidor Liberty.
 - a. Abra o Web Browser de dentro da VM e vá até a URL abaixo para exibir a página web do aplicativo principal.

```
http://localhost:9080
```



3. Faça uma pequena alteração no código-fonte do "**system Properties Sample**" enquanto rode no modo dev, para ver as alterações dinamicamente captadas e aplicadas ao servidor em execução.

a. Clique em **Activities** para acessar a Barra de Ferramentas, depois clique com o botão direito do mouse em Terminal para abrir uma nova janela do terminal.



b. Na nova janela do terminal, navegue até a pasta **src/main/webapp** do aplicativo conforme ilustrado abaixo.

```
cd /home/techzone/Student/labs/devmode/start/src/main/webapp
```

b. Use o editor do gedit para abrir o arquivo **index.html** no modo de edição

```
gedit index.html
```

c. Modifique a página index.html na linha número 25.

Alterar título na linha destacada:

```
16 <html>
17   <head>
18     <script src="js/mpData.js"></script>
19     <link href="https://fonts.googleapis.com/css?family=Asap" rel="stylesheet">
20     <link rel="stylesheet" href="css/main.css">
21   </head>
22   <body>
23     <section id="appIntro">
24       <div id="titleSection">
25         <h1 id="appTitle">System Properties Sample</h1>
26         <div class="line"></div>
27         <div class="headerImage"></div>
28       <h2>Congrats on your shiny, new Open Liberty sample app!</h2>
```


Título atualizado: Demo de Propriedades do Sistema

```
16 <html>
17   <head>
18     <script src="js/mpData.js"></script>
19     <link href="https://fonts.googleapis.com/css?family=Asap" rel="stylesheet">
20     <link rel="stylesheet" href="css/main.css">
21   </head>
22   <body>
23     <section id="appIntro">
24       <div id="titleSection">
25         <h1 id="appTitle">System Properties Demo</h1>
26         <div class="line"></div>
27         <div class="headerImage"></div>
28       <h2>Congrats on your shiny, new Open Liberty sample app!</h2>
```

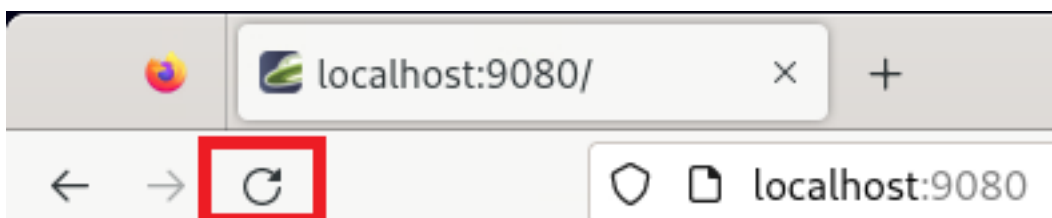
d. Salve o arquivo e feche o editor do gedit.

4. Acesse o microserviço **System Properties Sample** que foi implementado no servidor Liberty.

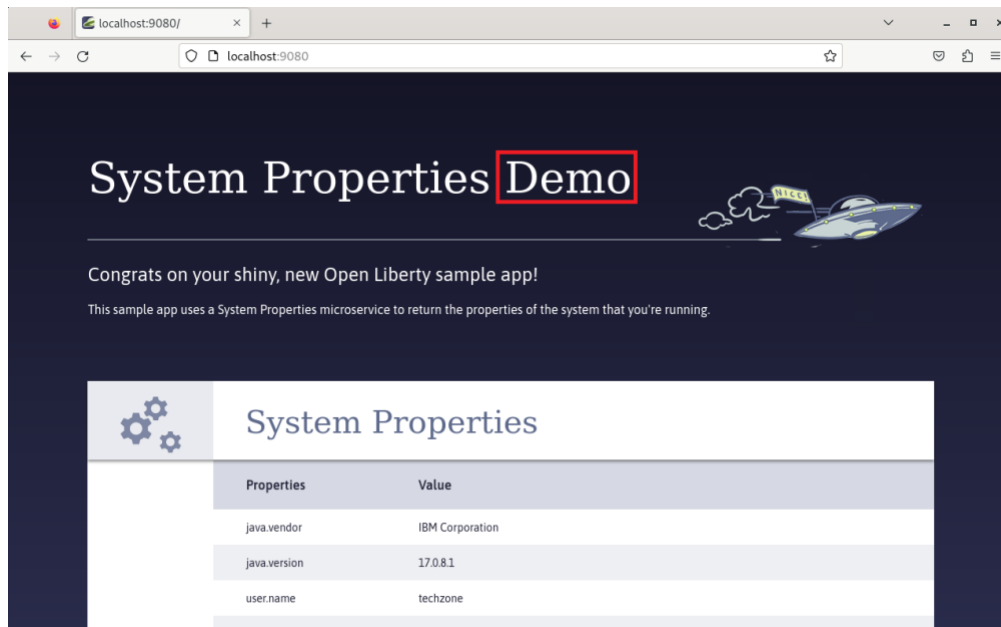
a. Acesse o Web Browser dentro da VM, e navegue na URL abaixo para exibir a página web do aplicativo principal.

`http://localhost:9080`

- c. **IMPORTANTE:** Clique no ícone RELOAD no navegador para recarregar a página. Conteúdo de cache de navegadores, portanto, é necessário recarregar a página.



- c. A página index.html atualizada é exibida com suas alterações dinamicamente captadas



As alterações do código de aplicação foram detectadas e aplicadas dinamicamente na instância de execução do servidor Liberty.

2.3 Como atualizar a configuração do Servidor sem reiniciar o servidor

O plugue do Open Liberty Maven plug-in não só atende a mudanças de código de aplicativos, mas também mudanças de configuração no projeto. O servidor Open Liberty recarrega automaticamente a configuração sem reiniciar. Esse objetivo permite um turnarounds mais rápido e uma experiência de desenvolvedor melhorada.

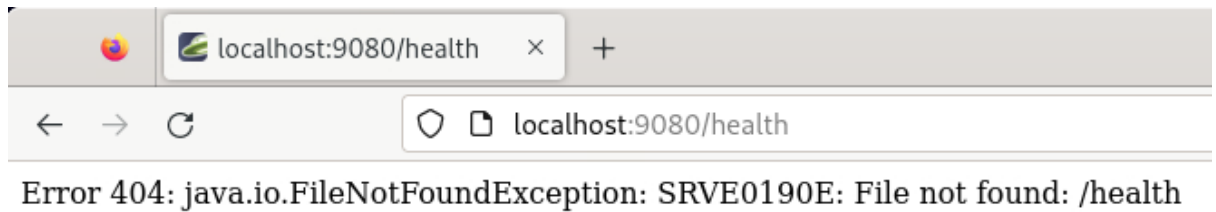
Nesta seção, você faz uma alteração de configuração simples no arquivo de configuração do Liberty Server (server.xml) do projeto (src), para incluir um endpoint **/health** para o serviço. Você notará que as alterações foram detectadas pelo maven e atualizadas dinamicamente no **server.xml** de destino. A instância Liberty em execução escolhe automaticamente a configuração de destino alterada

Se você tentar acessar este terminal **/health** agora, você vê um erro de 404 porque o terminal **/health** ainda não existe:

1. A partir do navegador da Web na VM, tente acessar o terminal de aplicativos **/health** em:

```
http://localhost:9080/health
```

Nota: você vê um erro de 404 porque o terminal **/health** ainda não existe



2. Faça uma alteração simples no arquivo de configuração do Liberty Server para adicionar o recurso **mpHealth-4.0** no arquivo **server.xml**, que possibilita o terminal de verificação de funcionamento.

a A partir de uma janela do terminal, navegue até a pasta liberty/config do aplicativo

```
cd  
/home/techzone/Student/labs/devmode/start/src/main/liberty/config
```

b. Use o editor do gedit para abrir o arquivo server.xml no modo de edição

```
gedit server.xml
```

c. Faça a seguinte alteração no arquivo **server.xml**

Inclua uma linha na seção featureManager:

```
2    <featureManager>  
3        <feature>restfulWS-3.1</feature>  
4        <feature>jsonp-2.1</feature>  
5        <feature>jsonb-3.0</feature>  
6        <feature>cdi-4.0</feature>  
7        <feature>mpMetrics-5.1</feature>  
8        <feature>mpConfig-3.1</feature>  
9    </featureManager>
```

Atualize para incluir: <feature>mpHealth-4.0</feature>

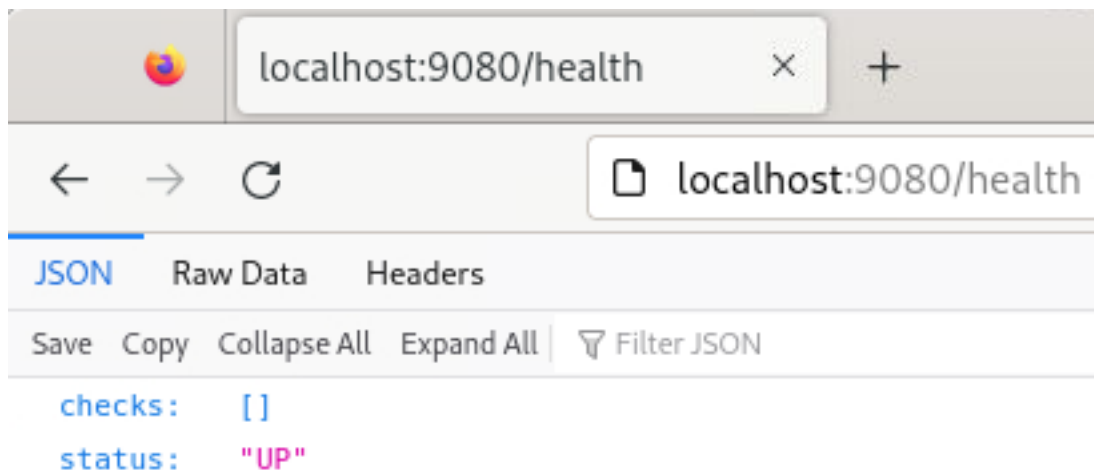
```
2    <featureManager>
3        <feature>restfulWS-3.1</feature>
4        <feature>jsonp-2.1</feature>
5        <feature>jsonb-3.0</feature>
6        <feature>cdi-4.0</feature>
7        <feature>mpMetrics-5.1</feature>
8        <feature>mpConfig-3.1</feature>
9        <feature>mpHealth-4.0</feature>
10    </featureManager>
```

d. Salve o arquivo e feche o editor do gedit.

3. Acesse o novo **endpoint Health** para a aplicação.

a. Abra o Web Browser de dentro da VM e vá até a URL abaixo para exibir o terminal de saúde.

```
http://localhost:9080/health
```



4. Visualize o log do console do servidor Liberty para ver se o recurso **mpHealth-4.0** foi instalado, e o endpoint health foi ativado.

a. Retorne para a janela do terminal onde a `mvn liberty:dev` está em execução.

b. Analise as mensagens que indicam o recurso `mpHealth-4.0` foi instalado dinamicamente e o endpoint `/health` ativado.

```

[INFO] [WARNING ] SRVE0190E: File not found: /health
[INFO] Copied file: /home/techzone/Student/labs/devmode/start/src/main/liberty/config/server.xml to: /tmp/tempConfig11139426987896733758/server.xml
[INFO] Configuration features have been added: [mpHealth-4.0]
[INFO] Running liberty:install-feature
[INFO] Feature signature verify option: enforce
[INFO] Parsing the server file for features and includes: tempConfig11139426987896733758/server.xml
[INFO] Parsing the server file for features and includes: tempConfig11139426987896733758/configDropins/overrides/liberty-plugin-variable-config.xml
[INFO] plugin listed esa: []

[INFO] Resolving features...
[INFO] Downloading public key(s) for signature verification
[INFO] Verifying features
[INFO] Verifying signatures ...
<-----> 0.00%
All features were successfully verified.
All features were successfully verified.
[INFO] Installing features: [mpconfig-3.1, jsonb-3.0, mpHealth-4.0, restfulws-3.1, jsonp-2.1, mpmetrics-5.1, cdi-4.0]
[INFO] Product validation completed successfully.
[INFO] The following features have been installed: mpHealth-4.0
[INFO] Copied file: /home/techzone/Student/labs/devmode/start/src/main/liberty/config/server.xml to: /home/techzone/Student/labs/devmode/start/target/liberty/wlp/usr/servers/defaultServer/server.xml
[INFO] Running liberty:deploy
[INFO] Copying 1 file to /home/techzone/Student/labs/devmode/start/target/liberty/wlp/usr/servers/defaultServer
[INFO] CWWKM2144I: Update server configuration file server.xml from /home/techzone/Student/labs/devmode/start/src/main/liberty/config/server.xml.
[INFO] CWWKM2185I: The liberty-maven-plugin configuration parameter "appsDirectory" value defaults to "apps".
[INFO] Application configuration is found in server.xml : guide-getting-started.war
[INFO] CWWKM2160I: Installing application guide-getting-started.war.xml.
[INFO] CWWKM2010I: Searching for CWWKZ0001I.*guide-getting-started in /home/techzone/Student/labs/devmode/start/target/liberty/wlp/usr/servers/defaultServer/logs/messages.log. This search will timeout after 40 seconds.
[INFO] CWWKM2015I: Match number: 1 is [7/30/24, 6:36:18:164 EDT] 00000037 com.ibm.ws.app.manager.AppMessageHelper A CWWKZ0001I:
Application guide-getting-started started in 5.816 seconds..
[INFO] [AUDIT ] CWWKG0016I: Starting server configuration update.
[INFO] [AUDIT ] CWWKG0093A: Processing configuration drop-ins resource: /home/techzone/Student/labs/devmode/start/target/liberty/wlp/usr/servers/defaultServer/configDropins/overrides/liberty-plugin-variable-config.xml
[INFO] [AUDIT ] CWWKT0017I: Web application removed (default_host): http://rhel9-base.gym.lan:9080/
[INFO] [AUDIT ] CWWKZ0009I: The application guide-getting-started has stopped successfully.
[INFO] [AUDIT ] CWWKG0017I: The server configuration was successfully updated in 0.932 seconds.
[INFO] [AUDIT ] CWWKT0016I: Web application available (default_host): http://rhel9-base.gym.lan:9080/health/
[INFO] [AUDIT ] CWWKF1037I: The server added the [mpHealth-4.0] features to the existing feature set.
[INFO] [AUDIT ] CWWKF0012I: The server installed the following features: [cdi-4.0, distributedMap-1.0, jndi-1.0, json-1.0, jsonb-3.0, jsonp-2.1, monitor-1.0, mpConfig-3.1, mpHealth-4.0, mpMetrics-5.1, restfulWS-3.1, restfulWSClient-3.1, ssl-1.0, transportSecurity-1.0].
[INFO] [AUDIT ] CWWKF0008I: Feature update completed in 1.120 seconds.
[INFO] [AUDIT ] CWWKT0016I: Web application available (default_host): http://rhel9-base.gym.lan:9080/
[INFO] [AUDIT ] CWWKZ0003I: The application guide-getting-started updated in 0.779 seconds.

```

As alterações de configuração do servidor Liberty foram detectadas e aplicadas dinamicamente na instância de execução do servidor Liberty.

O endpoint /health informa se o servidor está em execução, mas o terminal não fornece nenhum detalhe sobre os microserviços que estão em execução dentro do servidor.

O MicroProfile Health oferece verificações de readiness and liveness.

- **Readiness** permite que serviços de terceiros, como o de Kubernetes, saibam se o microserviço está pronto para processar pedidos.
- **Liveness** permite que serviços de terceiros determinem se o microserviço está em execução.

5. A partir da janela do terminal que está em execução "mvn:liberty:dev", use CTL-C para parar o servidor Liberty.

2.4 Desenvolvendo e Executando o Aplicativo em um Container Docker e no Liberty Dev Mode

Ao desenvolver um aplicativo que acabará sendo implementado para a produção em contêineres, você pode evitar problemas potenciais garantindo que seus ambientes de desenvolvimento e produção sejam o mais semelhantes possível. Isto alinha com a metodologia [Twelve Factor App](#), particularmente fator 10, que pede paridade de dev / prod. Para aplicações nativas em nuvem, parte desta questão é abordada por meio do uso de

contêineres onde seu ambiente pode ser codificado para proporcionar consistência entre desenvolvimento e produção.

Nesta seção do laboratório, você usará o modo Liberty dev com contêineres. Com o suporte de contêiner, você pode desenvolver aplicativos em seu ambiente local enquanto seu servidor Open Liberty é executado em um contêiner.

A imagem do container de desenvolvimento é mantida o mais semelhante possível para a imagem de container de produção, enquanto ainda permite o desenvolvimento iterativo. Suas alterações de código são automaticamente quentes implementadas no contêiner e captadas pelo servidor em execução. Adicionalmente, o modo dev permite que você execute testes automaticamente ou sob demanda, e você pode anexar um depurador a qualquer momento para depurar sua aplicação.

Com o suporte de container para o modo Open Liberty dev, você pode usar o mesmo Dockerfile tanto para o desenvolvimento quanto para a produção. Você usa a mesma imagem de base e customizações, e especifica os arquivos de configuração exatos que você precisa para sua aplicação em seu Dockerfile. Isso evita que você entre em qualquer surpresa ao implantar sua aplicação na produção.

O modo dev altera como a imagem é construída e executada para permitir o desenvolvimento iterativo, portanto as imagens são idênticas, exceto para como os arquivos de aplicação e de configuração são montados nos contêineres. Com o modo dev, basta salvar um arquivo de origem em qualquer editor de texto ou IDE, e ele é recompilado e captado sem necessidade de reconstruir a imagem ou reiniciar o servidor.

2.5 Investigar Comandos Docker e Dockerfile para construção de imagens

Esta seção aborda como utilizar, construir, implementar e executar o Liberty em um contêiner Docker. Você irá puxar o Liberty do Docker hub, instalá-lo, revisar informações sobre o container, acessar recursos dentro do contêiner, adicionar um aplicativo e testá-lo.

Para executar o aplicativo em um contêiner, o Docker precisa ser instalado e o daemon do Docker em execução. Neste ambiente de laboratório, esses pré-requisitos foram configurados.

1. A partir da janela do terminal, utilize o CTL-C para parar o servidor Liberty, caso ainda esteja em execução a partir da seção anterior.
2. Na janela do Terminal, verifique se o Docker está rodando

```
docker --version
```



```
[techzone@rhel9-base start]$ docker --version
Docker version 24.0.7, build afdd53b
[techzone@rhel9-base start]$
```

3. Execute o exemplo docker **hello-world**. A imagem do docker será puxada a partir do Dockerhub se ele já não estiver na máquina local. Se Docker estiver funcionando adequadamente, você verá a mensagem destacada ilustrada abaixo, informando que Docker está trabalhando corretamente.

```
docker run --name hello-world hello-world
```

```
techzone@rhel9-base:~/Student/labs/devmode/start
[techzone@rhel9-base start]$ docker run --name hello-world hello-world
Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/
```

4. Use o comando **docker images** para descobrir quais imagens do Docker estão no repositório do docker local

```
docker images hello-world
```

```
[techzone@rhel9-base start]$ docker images hello-world
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
hello-world   latest    9c7a54a9a43c   15 months ago 13.3kB
```

5. Use o comando **docker history** para visualizar as camadas que compõem a imagem do Docker

```
docker history hello-world
```

```
[techzone@rhel9-base start]$ docker history hello-world
IMAGE          CREATED          CREATED BY          SIZE      COMMENT
9c7a54a9a43c   15 months ago   /bin/sh -c #(nop)  CMD ["/hello"]     0B
<missing>      15 months ago   /bin/sh -c #(nop)  COPY file:201f8f1849e89d53... 13.3kB
```

6. Vamos executar um contêiner Docker de imagem Liberty. Docker irá verificar se há uma imagem no repositório. Se não, ele fará o download da imagem mais recente, em seguida, execute-a

```
docker run -d -p 9086:9080 --name wlp websphere-liberty
```

```
[techzone@rhel9-base start]$ docker run -d -p 9086:9080 --name wlp websphere-liberty
Unable to find image 'websphere-liberty:latest' locally
latest: Pulling from library/websphere-liberty
3713021b0277: Pull complete
238ad37218bc: Pull complete
e1ba472852db: Pull complete
ccf32fbb3b06: Pull complete
7bac481e4a3c: Pull complete
5e62cebd93a2: Pull complete
c0969a581acd: Pull complete
cf827c50e11c: Pull complete
f0f4549f557f: Pull complete
0eda2b98e513: Pull complete
9c47delaec6d: Pull complete
8e92b98f0024: Pull complete
398b2ba226ef: Pull complete
70be1a42ab15: Pull complete
Digest: sha256:2aa46c3be4d51d056867ed70f93e8d2bbc28ef92bfde74715b242fa2d16bfc81
Status: Downloaded newer image for websphere-liberty:latest
89346ed90ad291eca8b70c675e55a3134d576cc4fd9da101388c1f32488d1b18
```

7. Revise as informações de containers.

- a. O comando **docker ps** lista apenas contêineres em execução. O comando **docker ps -a** mostra todos os contêineres, em execução ou parada.

```
docker ps | grep liberty
```

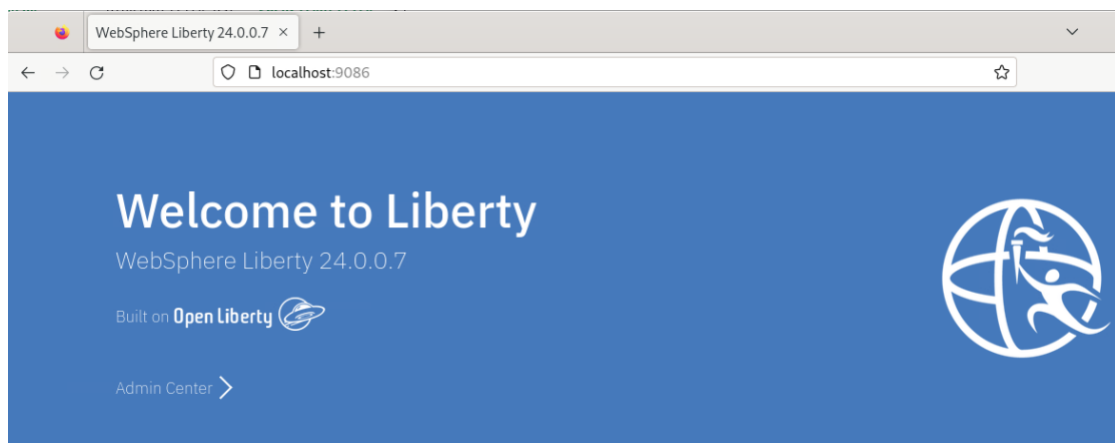
```
[techzone@rhel9-base start]$ docker ps | grep liberty
89346ed90ad2 websphere-liberty "/opt/ibm/helpers/ru..." About a minute ago Up About a minute
9443/tcp, 0.0.0.0:9086->9080/tcp, :::9086->9080/tcp wlp
```

- c. O comando **docker stats** mostra o uso de recursos dos contêineres em execução. Use as teclas Ctrl + C para parar o comando stats.

```
docker stats
```

techzone@rhel9-base:~/Student/labs/devmode/start — docker stats							
CONTAINER ID	NAME	CPU %	MEM USAGE / LIMIT	MEM %	NET I/O	BLOCK I/O	PIDS
89346ed90ad2	wlp	1.09%	237.5MiB / 7.463GiB	3.11%	2.84kB / 286B	91.3MB / 26.8MB	70

8. Abra o navegador da Web na VM e acesse o Liberty em execução no contêiner:
<http://localhost:9086>



9. Veja os logs do Liberty no contêiner em execução

```
docker logs wlp
```

```
techzone@rhel9-base:~/Student/labs/devmode/start
[techzone@rhel9-base start]$ docker logs wlp

Launching defaultServer (WebSphere Application Server 24.0.0.7/wlp-1.0.91.cl240720240701-1102) on IBM J9 VM
, version 8.0.8.26 - pxa6480sr8fp26-20240529_01(SR8 FP26) (en_US)
[AUDIT ] CWWKE0001I: The server defaultServer has been launched.
[AUDIT ] CWWKE0100I: This product is licensed for development, and limited production use. The full licen
se terms can be viewed here: https://public.dhe.ibm.com/ibmdl/export/pub/software/websphere/wasdev/license/
base_ilan/ilan/24.0.0.7/lafiles/en.html
[AUDIT ] CWWKG0093A: Processing configuration drop-ins resource: /opt/ibm/wlp/usr/servers/defaultServer/c
onfigDropins/defaults/keystore.xml
[WARNING ] CWWKS3103W: There are no users defined for the BasicRegistry configuration of ID com.ibm.ws.secu
rity.registry.basic.config[basic].
[AUDIT ] CWWKZ0058I: Monitoring dropins for applications.
[AUDIT ] CWWKS4104A: LTPA keys created in 0.770 seconds. LTPA key file: /opt/ibm/wlp/output/defaultServer
/resources/security/ltpa.keys
[AUDIT ] CWWKT0016I: Web application available (default_host): http://89346ed90ad2:9080/ibm/api/
[AUDIT ] CWWKT0016I: Web application available (default_host): http://89346ed90ad2:9080/openapi/
[AUDIT ] CWWKT0016I: Web application available (default_host): http://89346ed90ad2:9080/health/
[AUDIT ] CWWKT0016I: Web application available (default_host): http://89346ed90ad2:9080/metrics/
[AUDIT ] CWWKT0016I: Web application available (default_host): http://89346ed90ad2:9080/openapi/ui/
[AUDIT ] CWWKT0016I: Web application available (default_host): http://89346ed90ad2:9080/jwt/
[AUDIT ] CWWKF0012I: The server installed the following features: [appClientSupport-1.0, appSecurity-2.0,
appSecurity-3.0, batch-1.0, beanValidation-2.0, cdi-2.0, concurrent-1.0, distributedMap-1.0, ejb-3.2, ejbH
ome-3.2, ejbLite-3.2, ejbPersistentTimer-3.2, ejbRemote-3.2, el-3.0, j2eeManagement-1.1, jacc-1.5, jaspic-1
.1, javaMail-1.6, javaee-8.0, jaxb-2.2, jaxrs-2.1, jaxrsClient-2.1, jaxws-2.2, jca-1.7, jcaInboundSecurity-
1.0, jdbc-4.2, jms-2.0, jndi-1.0, jpa-2.2, jpaContainer-2.2, jsf-2.3, json-1.0, jsonb-1.0, jsonp-1.1, jsp-2
.3, jwt-1.0, managedBeans-1.0, mdb-3.2, microProfile-3.0, mpConfig-1.3, mpFaultTolerance-2.0, mpHealth-2.0,
mpJwt-1.1, mpMetrics-2.0, mpOpenAPI-1.1, mpOpenTracing-1.3, mpRestClient-1.3, opentracing-1.3, servlet-4.0
, ssl-1.0, wasJmsClient-2.0, wasJmsSecurity-1.0, wasJmsServer-1.0, webProfile-8.0, websocket-1.1].
[AUDIT ] CWWKF0011I: The defaultServer server is ready to run a smarter planet. The defaultServer server
started in 4.701 seconds.
[AUDIT ] CWWPI0803A: SSL certificate created in 2.300 seconds. SSL key file: /opt/ibm/wlp/output/defaultS
erver/resources/security/key.p12
[AUDIT ] CWWKI0001I: The CORBA name server is now available at corbaloc:iiop:localhost:2809/NameService.
```

10. Pare e remova os contêineres de docker usados nesta seção do laboratório. Em seguida, use o docker ps -a comando para verificar os contêineres "wlp" e "hello-mundo" são removidos.

```
docker stop wlp
docker rm wlp
docker rm hello-world
docker ps -a
```

2.6 Executando o aplicativo em um contêiner

Para executar o aplicativo em um contêiner, o Docker precisa ser instalado e o daemon do Docker em execução. Neste ambiente de laboratório, esses pré-requisitos foram configurados.

Primeiro, para confinar o aplicativo, você precisa de um **Dockerfile**. Este arquivo contém uma coleção de instruções que definem como uma imagem do Docker é construída, quais arquivos são empacotados nele, quais comandos são executados quando a imagem é executada como um contêiner, e outras informações.

Para este laboratório, foi fornecido um **Dockerfile** para construir a imagem do docker para a Amostra de Propriedades do Sistema. Este Dockerfile copia o arquivo **.war** em uma imagem Docker que contém o tempo de execução Java e um servidor Open Liberty pré-configurado.

1. A partir de uma janela do terminal, pare o Liberty Server em execução a partir da seção anterior do laboratório, utilizando os comandos abaixo:

```
cd /home/techzone/Student/labs/devmode/start
mvn liberty:stop
```

2. Visualize o **Dockerfile** que é usado para construir a imagem do docker.

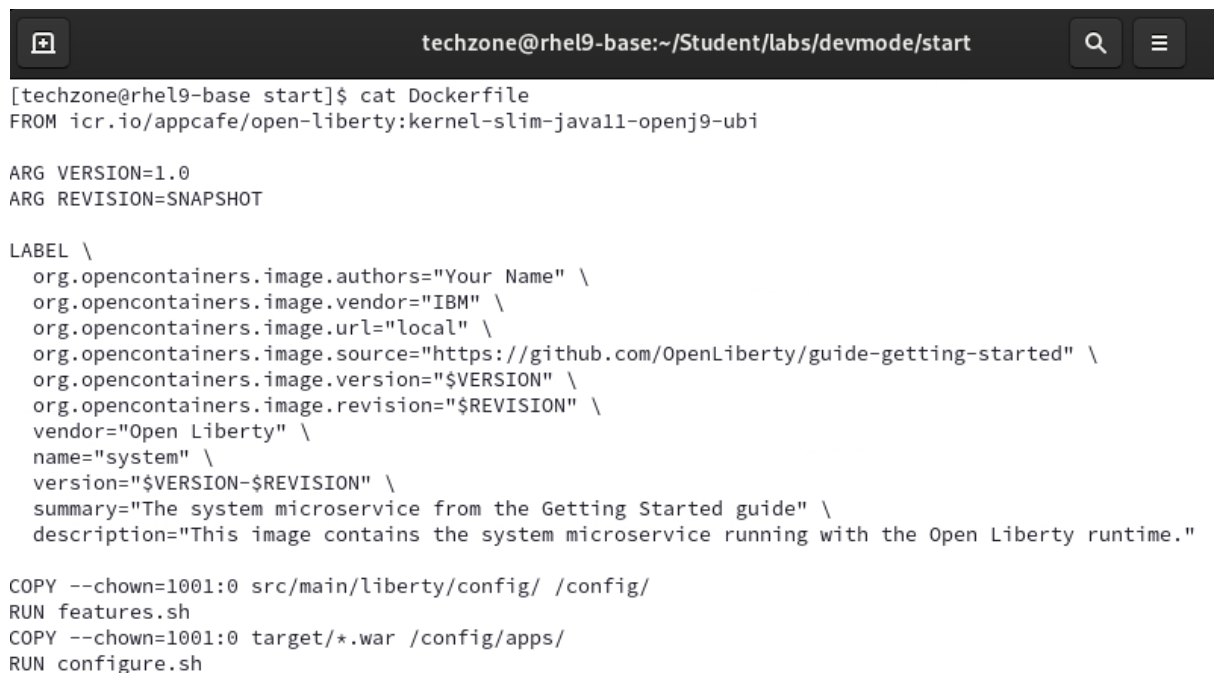
- a. Abra uma janela de terminal e mude para o diretório: /home/techzone/Student/labs/devmode/start

```
cd /home/techzone/Student/labs/devmode/start
```

- b. Investigue o Dockerfile

```
cat Dockerfile
```

O Dockerfile executa as seguintes tarefas:



```
techzone@rhel9-base:~/Student/labs/devmode/start
[techzone@rhel9-base start]$ cat Dockerfile
FROM icr.io/appcafe/open-liberty:kernel-slim-java11-openj9-ubi

ARG VERSION=1.0
ARG REVISION=SNAPSHOT

LABEL \
  org.opencontainers.image.authors="Your Name" \
  org.opencontainers.image.vendor="IBM" \
  org.opencontainers.image.url="local" \
  org.opencontainers.image.source="https://github.com/OpenLiberty/guide-getting-started" \
  org.opencontainers.image.version="$VERSION" \
  org.opencontainers.image.revision="$REVISION" \
  vendor="Open Liberty" \
  name="system" \
  version="$VERSION-$REVISION" \
  summary="The system microservice from the Getting Started guide" \
  description="This image contains the system microservice running with the Open Liberty runtime."

COPY --chown=1001:0 src/main/liberty/config/ /config/
RUN features.sh
COPY --chown=1001:0 target/*.war /config/apps/
RUN configure.sh
```

Leia abaixo uma descrição curta e detalhes dos comandos no Dockerfile:

- **FROM openliberty/open-liberty:kernel-java11-openj9-ubi**

FROM especifica a imagem do Docker que deve ser utilizada. Se isso não estiver no repositório local, este será puxado do Docker Hub.

Neste exemplo, puxamos a imagem do kernel liberty que contém Java 11, e usa o RedHat Universal base Images para implementações ao OpenShift.

A imagem do kernel contém apenas o kernel Liberty e sem recursos de tempo de execução adicionais. Esta imagem é a base recomendada para imagens construídas personalizadas, de modo que elas possam conter apenas os recursos necessários para uma aplicação específica

- **ARG VERSION=1.0 e ARG REVISION=SNAPSHOT**

A instrução ARG define variáveis que podem ser passadas no tempo de construção. Uma vez definido no Dockerfile, você pode passá-lo com a flag -- build-arg.

- **LABEL**

Labels são usados no Dockerfile para ajudar a organizar o seu Docker Images. Labels são pares de chave valor e simplesmente adicionam metadados personalizados ao seu Docker Images.

- **COPY -- chown=1001: 0 src/main/liberty/config/ /config/**

O comando COPY irá copiar o arquivo de configuração do liberty (server.xml) para a pasta /config no contêiner Liberty.

As Imagens de Base Universal RedHat (UBI) são construídas de tal forma que a Liberty não é executada como raiz. Para garantir que o Liberty possa acessar os arquivos que são copiados para a imagem, eles devem ser de propriedade de um usuário não root.

Todas as imagens Liberty da IBM contêm um usuário não root definido como 1001:0. Assim, o comando copy copia os arquivos como o usuário não root que é conhecido por existir na imagem LIBERTY.

- **RUN features.sh**

O script features.sh adicionará os trechos XML solicitados para ativar os recursos Liberty e fazer crescer a imagem para ser apto-para-propósito usando featureUtility.

- **COPY -- chown=1001: 0 target / *.war /config/apps/**

O comando COPY irá copiar o arquivo WAR do aplicativo para o diretório /config/apps sobre a imagem.

- **RUN configure.sh**

O script configure.sh adicionará as configurações do servidor solicitadas, aplicará quaisquer correções provisórias e preencherá caches para otimizar o tempo de execução.

3. Execute o comando do pacote mvn a partir do diretório inicial. O comando produzirá um arquivo WAR chamado " **guide-getting-started.war**" e copiado para o diretório "target".

```
mvn package
```

```
[techzone@rhel9-base start]$ mvn package
[INFO] Scanning for projects...
[INFO]
[INFO] -----< io.openliberty.guides:guide-getting-started >-----
[INFO] Building guide-getting-started 1.0-SNAPSHOT
[INFO]    from pom.xml
[INFO] -----[ war ]-----
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-war-plugin/3.1.0/maven-war-plugin-3.1.0.jar
```

O comando do pacote Maven constrói o aplicativo e produz um arquivo binário implementável de aplicativo ". war" que é copiado na imagem do Docker através do Dockerfile.

4. Execute o seguinte comando para fazer o download ou atualização para a mais recente imagem do Open Liberty Docker usada em nosso Dockerfile:

```
docker pull icr.io/appcafe/open-liberty:kernel-slim-java11-openj9-ubi
```

Observação: Se a imagem já foi puxada para esta VM, você poderá ver apenas uma mensagem indicando que a imagem já está atualizada, em vez de puxar a imagem novamente.

```
[techzone@rhel9-base start]$ docker pull icr.io/appcafe/open-liberty:kernel-slim-javall-openj9-ubi
kernel-slim-javall-openj9-ubi: Pulling from appcafe/open-liberty
6468b01f0a9a: Already exists
dda1bb03ce5d: Already exists
c90fb4b96425: Already exists
0f12c9569ac9: Already exists
c7948d1f56b9: Already exists
2f2de7821029: Already exists
f6cc21928397: Already exists
58b2bf5b5dc1: Already exists
34e58f26a622: Already exists
ddeb0b957b8a: Already exists
259228c50ff7: Already exists
06124f44d6dd: Already exists
0f66fb7befff: Already exists
4e55adbc9194: Already exists
6c0207342050: Already exists
b2ad2d57424c: Already exists
b42ad8f9992c: Already exists
Digest: sha256:cdb1c976f0c6bdf78a5f1083c09ae338fbb56d22e2f65bd420699f15fef09cf
Status: Downloaded newer image for icr.io/appcafe/open-liberty:kernel-slim-javall-openj9-ubi
icr.io/appcafe/open-liberty:kernel-slim-javall-openj9-ubi
```

OU

```
[techzone@rhel9-base start]$ docker pull icr.io/appcafe/open-liberty:kernel-slim-javall-openj9-ubi
kernel-slim-javall-openj9-ubi: Pulling from appcafe/open-liberty
Digest: sha256:cdb1c976f0c6bdf78a5f1083c09ae338fbb56d22e2f65bd420699f15fef09cf
Status: Image is up to date for icr.io/appcafe/open-liberty:kernel-slim-javall-openj9-ubi
icr.io/appcafe/open-liberty:kernel-slim-javall-openj9-ubi
```

5. Para construir e containerizar a aplicação, execute o comando de construção do Docker a seguir. Tenha certeza que você está no diretório que contém o arquivo Docker.

Observação: O ponto no final do comando de construção do docker faz parte do comando, indicando utilizar o caminho do diretório atual.

```
cd /home/techzone/Student/labs/devmode/start
docker build -t openliberty-getting-started:1.0-SNAPSHOT .
```

```
[techzone@rhel9-base start]$ docker build -t openliberty-getting-started:1.0-SNAPSHOT .
[+] Building 24.4s (10/10) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 851B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for icr.io/appcafe/open-liberty:kernel-slim-javall-openj9-ubi
=> [1/5] FROM icr.io/appcafe/open-liberty:kernel-slim-javall-openj9-ubi
=> [internal] load build context
=> => transferring context: 265B
=> CACHED [2/5] COPY --chown=1001:0 src/main/liberty/config/ /config/
=> CACHED [3/5] RUN features.sh
=> CACHED [4/5] COPY --chown=1001:0 target/*.war /config/apps/
=> [5/5] RUN configure.sh
=> exporting to image
=> => exporting layers
=> => writing image sha256:7bcbbcd15d549576c084b65ad66fbb756673a13f3ee0b76fd15729fd63364f91
=> => naming to docker.io/library/openliberty-getting-started:1.0-SNAPSHOT
[techzone@rhel9-base start]$
```

A imagem do Docker **openliberty-getting-started:1.0-SNAPSHOT** é construída a partir do Dockerfile.

6. Para verificar se a imagem é construída, execute o comando `docker images` para listar todas as imagens do Docker local

```
docker images | grep getting-started
```

Sua imagem deve ser listada como "**openliberty-getting-started**" com o TAG de "1,0-SNAPSHOT"

```
[techzone@rhel9-base start]$ docker images | grep getting-started
openliberty-getting-started          1.0-SNAPSHOT          7bcbbcd15d54   12 minutes ago   757MB
```

7. Em seguida, execute o contêiner a partir da imagem.

```
docker run -d --name gettingstarted-app -p 9080:9080 openliberty-
getting-started:1.0-SNAPSHOT
```

```
[techzone@rhel9-base start]$ docker run -d --name gettingstarted-app -p 9080:9080 openliberty-getting-started:1.0-SNAPSHOT
d3e244c0b7f30e1b552295e0ac103a681039d4fbcbf0439b5a4a6e1ab8ec2306
```

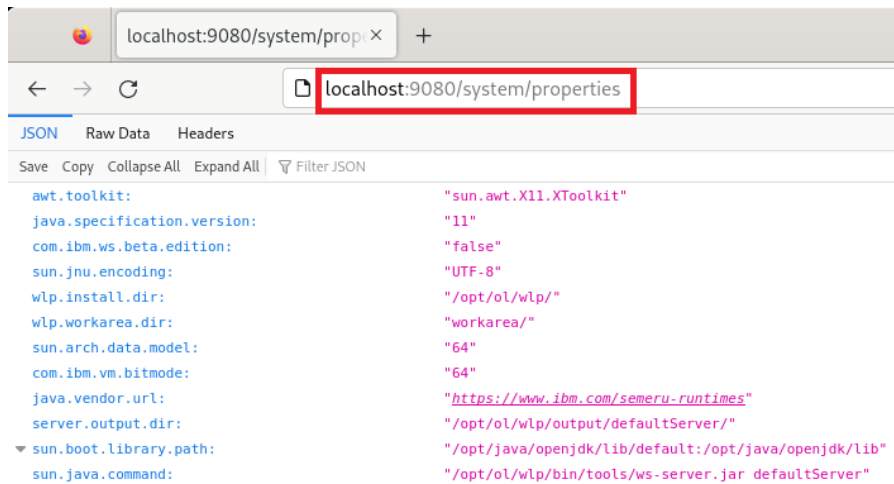
8. Execute o comando `docker ps` para verificar o seu contêiner de docker está em execução

```
docker ps -l
```

```
[techzone@rhel9-base start]$ docker ps -l
```

CONTAINER ID	IMAGE	NAMES	COMMAND	CREATED	STATUS	PORTS
d3e244c0b7f3	openliberty-getting-started:1.0-SNAPSHOT		"/opt/ol/helpers/run..."	3 minutes ago	Up 3 minutes	0.0.0.0:9080->9080/tcp, :::9080->9080/tcp, 9443/tcp
		gettingstarted-app				

9. A partir de um Web Browser, acesse o aplicativo usando: <http://localhost:9080/system/properties>



10. Parar e Remover o contêiner Docker

```
docker stop gettingstarted-app  
docker rm gettingstarted-app
```

11. Remover a imagem do Docker

```
docker rmi openliberty-getting-started:1.0-SNAPSHOT
```

Parabéns! Você concluiu com sucesso o laboratório " Introdução ao Liberty Modo Dev.