Usando JHipster en producción

JHipster genera una aplicación totalmente lista para producción, optimizada y segura. Esta sección describe las opciones más importantes: si tiene prisa, simplemente ejecute una compilación de producción normal, ¡pero no olvide leer la sección de seguridad!

Construyendo un paquete de producción

Probar una compilación de producción

Esto permite probar una compilación de producción de Maven, sin construir un paquete real.

Para usar JHipster en modo "producción", use el `prod` perfil preconfigurado . Con Maven, ejecuta:

```
./mvnw -Pprod
```

Cuando use Gradle, ejecute:

```
./gradlew -Pprod
```

Este perfil compilará, probará y empaquetará su aplicación con todas las configuraciones de producción.

Si desea más información sobre los perfiles disponibles, vaya a la sección titulada " Perfiles de desarrollo y producción (https://www.jhipster.tech/profiles/) ".

Construyendo un archivo JAR / WAR ejecutable

Para empaquetar la aplicación como un JAR de "producción", con Maven, escriba:

```
./mvnw -Pprod clean verify
```

O cuando use Gradle, escriba:

```
./gradlew -Pprod clean bootJar
```

Esto generará este archivo (si su aplicación se llama "jhipster"):

Cuando use Maven:

- `target/jhipster-0.0.1-SNAPSHOT.jar`

Cuando use Gradle:

- `build/libs/jhipster-0.0.1-SNAPSHOT.jar`

Para empaquetar la aplicación como WAR de “producción”, con Maven, escriba:

```
./mvnw -Pprod,war clean verify
```

O cuando use Gradle, escriba:

```
./gradlew -Pprod -Pwar clean bootWar
```

Tenga en cuenta que al compilar un archivo JAR o WAR con una ruta de contexto, deberá actualizar webpack.prod.js con la baseHref adecuada.

Esto generará estos archivos (si su aplicación se llama "jhipster"):

Cuando use Maven:

- `target/jhipster-0.0.1-SNAPSHOT.war`

Cuando use Gradle:

- `build/libs/jhipster-0.0.1-SNAPSHOT.war`

Tenga en cuenta que al crear un archivo JAR o WAR con el `prod` perfil, el archivo generado no incluirá los `dev` activos.

Tenga en cuenta que si desea el archivo original WAR con Maven, debe editar el `pom.xml` archivo para usar el `war` paquete en lugar del `jar` paquete:

```
-    <packaging>jar</packaging>
+    <packaging>war</packaging>
```

Corriendo en producción

Ejecutar el archivo JAR sin un servidor de aplicaciones

En lugar de implementar en un servidor de aplicaciones, a muchas personas les resulta más fácil tener un archivo JAR ejecutable.

Con el archivo JAR generado en el paso anterior, puede ejecutarlo en modo "producción" escribiendo (en Mac OS X o Linux):

```
./jhipster-0.0.1-SNAPSHOT.jar
```

Si está en Windows, use:

```
java -jar jhipster-0.0.1-SNAPSHOT.jar
```

Tenga en cuenta que este archivo JAR utiliza el perfil que seleccionamos al compilarlo. Como se creó con el `prod` archivo de la sección anterior, se ejecutará con el `prod` perfil.

Ejecutando la aplicación en un contenedor Docker

JHipster tiene soporte de primera clase para Docker: es muy fácil agrupar su archivo JAR ejecutable en una imagen de Docker y ejecutarlo dentro de Docker.

Para aprender a empaquetar su aplicación con Docker, lea nuestra documentación de Docker Compose (https://www.jhipster.tech/docker-compose/) .

Ejecutar como un servicio

También es posible ejecutar Jar como un servicio de Linux, y es posible que desee forzar su `pom.xml` archivo antes de empaquetarlo. Para hacerlo, agregue la siguiente propiedad dentro `<configuration>` del `spring-boot-maven-plugin` complemento.

```
<embeddedLaunchScriptProperties>
    <mode>service</mode>
</embeddedLaunchScriptProperties>
```

Luego, configure su init.d con:

```
ln -s jhipster-0.0.1-SNAPSHOT.jar /etc/init.d/jhipster
```

Asegure su aplicación con:

```
chown jhuser:jhuser jhipster-0.0.1-SNAPSHOT.jar sudo chattr +i your-app.jar
```

Considerando `jhuser` una cuenta de sistema operativo no root que ejecutará la aplicación, la aplicación se puede ejecutar de esta manera:

```
service jhipster start|stop|restart
```

Hay muchas otras opciones que puede encontrar en la documentación de Spring Boot (https://docs.spring.io/spring-boot/docs/current/reference/html/deployment-install.html) , incluidos más pasos de seguridad y servicio de Windows.

Ejecutar la aplicación bajo una ruta de contexto

Al implementar una aplicación JHipster en un servidor de aplicaciones o personalizar su ruta de contexto, es necesario establecer el `baseHref` valor en `webpack.common.js` o webpack.prod.js igual a la ruta de contexto esperada.

Optimizaciones de rendimiento

Ajuste de caché

Si seleccionó un proveedor de caché al generar su aplicación, JHipster lo configurará automáticamente.

Sin embargo, los valores predeterminados de la memoria caché son bastante bajos, por lo que la aplicación puede ejecutarse en hardware modesto, y estos valores deben ajustarse en función de los requisitos comerciales específicos de su aplicación.

Por favor lee:

- La documentación de JHipster "utilizando caché" (https://www.jhipster.tech/using-cache/) para obtener más información sobre el proveedor de almacenamiento en caché que ha seleccionado y cómo se puede ajustar
- La última sección sobre monitoreo , para que pueda ajustar su caché de acuerdo con el uso real de su aplicación

Soporte HTTP / 2

JHipster admite HTTP / 2 utilizando la `jhipster.http.version` propiedad, que está configurada en el `application-prod.yml` archivo.

Para habilitar HTTP / 2, debe:

- Conjunto `jhipster.http.version: V_2_0`
- Configure HTTPS (consulte la sección de seguridad de esta documentación), ya que los navegadores obligan a usar HTTPS con HTTP / 2

gzipping

Dentro de un archivo JAR ejecutable, que utiliza el `prod` perfil, JHipster configura la compresión GZip en sus recursos web.

Por defecto, la compresión funcionará en todos los recursos estáticos (HTML, CSS, JavaScript) y en todas las solicitudes REST. Puede obtener más información sobre esta configuración mirando las `server.compression.*` claves en las propiedades de la aplicación Spring Boot, configuradas en el `application-prod.yml` archivo.

Tenga en cuenta que gzipping se realiza mediante el servidor de aplicaciones, por lo que esta sección sólo se aplica si se utiliza la opción “JAR ejecutable” descrito anteriormente. Si ejecuta su aplicación en un servidor de aplicaciones externo, tendrá que configurar por separado.

Encabezados de caché

Con el `prod` perfil, JHipster configura un filtro de Servlet que coloca encabezados de caché HTTP específicos en sus recursos estáticos (JavaScript, CSS, fuentes ...) para que los navegadores y servidores proxy los almacenen en caché.

Generando una aplicación JavaScript optimizada con Webpack

Este paso se activa automáticamente cuando crea su proyecto con el `prod` perfil. Si desea ejecutarlo sin iniciar una compilación de Maven, ejecute:

```
npm run build
```

Esto usará Webpack (https://webpack.github.io/) para procesar todos sus recursos estáticos (CSS, TypeScript, HTML, JavaScript, imágenes ...) para generar una aplicación optimizada del lado del cliente.

Durante este proceso, Webpack compilará el código TypeScript en código JavaScript, y también generará mapas de origen, por lo que la aplicación del lado del cliente todavía se puede depurar.

Esos activos optimizados se generarán `target/classes/static` para Maven o `build/resources/main/static` Gradle, y se incluirán en su JAR de producción final.

Este código se servirá cuando ejecute la aplicación con el `prod` perfil.

Seguridad

Asegurar las cuentas de usuario y administrador predeterminadas

JHipster viene con algunos usuarios predeterminados generados para usted. En producción, ¡ **debes** cambiar esas contraseñas predeterminadas!

Siga nuestra documentación de seguridad (https://www.jhipster.tech/security/) para aprender cómo cambiar esas contraseñas y proteger su aplicación.

Soporte HTTPS

HTTPS se puede configurar directamente en su aplicación JHipster, o usando un proxy front-end específico.

Configuración HTTPS con JHipster

HTTPS se configura utilizando las `server.ssl` claves de configuración estándar de Spring Security en su `application-prod.yml` archivo.

Para habilitar SSL, genere un certificado usando:

```
keytool -genkey -alias <your-application> -storetype PKCS12 -keyalg RSA -keysize 2048 -ke
ystore keystore.p12 -validity 3650
```

También puede usar Let's Encrypt usando este tutorial (https://maximilian-boehm.com/hp2121/Create-a-Java-Keystore-JKS-from-Let-s-Encrypt-Certificates.htm) .

Luego, modifique las `server.ssl` propiedades para que su `application-prod.yml` configuración se vea así:

```
server:
    port: 443
    ssl:
        key-store: keystore.p12
        key-store-password: <your-password>
        keyStoreType: PKCS12
        keyAlias: <your-application>
        ciphers: TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256, TLS_ECDHE_ECDSA_WITH_AES_256_GC
M_SHA384, TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA, TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA, TLS
_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256, TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384, TLS_ECDHE_
RSA_WITH_AES_128_GCM_SHA256, TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384, TLS_ECDHE_RSA_WITH_AE
S_128_CBC_SHA, TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA, TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256,
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384, TLS_DHE_RSA_WITH_AES_128_GCM_SHA256, TLS_DHE_RSA_W
ITH_AES_256_GCM_SHA384, TLS_DHE_RSA_WITH_AES_128_CBC_SHA, TLS_DHE_RSA_WITH_AES_256_CBC_SH
A, TLS_DHE_RSA_WITH_AES_128_CBC_SHA256, TLS_DHE_RSA_WITH_AES_256_CBC_SHA256
        enabled-protocols: TLSv1.2
```

El conjunto de cifrados refuerza la seguridad al desactivar algunos cifrados SSL antiguos y en desuso, esta lista se probó con SSL Labs (https://www.ssllabs.com/ssltest/)

Una vez que la `server.ssl.ciphers` propiedad está habilitada, JHipster forzará el orden en Undertow con esta propiedad (verdadero por defecto): `jhipster.http.useUndertowUserCipherSuitesOrder`

El `enabled-protocols` desactiva los viejos protocolos SSL.

Luego, el toque final para lograr el secreto perfecto hacia adelante. Agregue el siguiente indicador al inicio de JVM:

```
-Djdk.tls.ephemeralDHKeySize=2048
```

Para probar su configuración, puede ir a SSL Labs (https://www.ssllabs.com/ssltest/) .

Si todo está bien, obtendrás A +

Configuración HTTPS con un proxy front-end

Hay muchas soluciones para configurar un proxy HTTPS front-end frente a una aplicación JHipster. Se describe aquí las 2 los más comunes.

Con una arquitectura de microservicio, puede usar el soporte Traefik de JHipster:

- Sigue nuestra documentación Traefik (https://www.jhipster.tech/traefik/) para configurar su arquitectura
- Siga la documentación del sitio web oficial de Traefik (https://docs.traefik.io/user-guide/examples/) para configurar HTTPS

Si prefiere usar el servidor HTTP Apache, puede configurarlo con Let's Encrypt:

- Instalar Apache y nos dejó Cifrar: `apt-get install -y apache2 python-certbot-apache`
- Configure Let's Encrypt: `certbot --apache -d <your-domain.com> --agree-tos -m <your-email> --redirect`
- Configure la renovación automática de certificados SSL: agregue `10 3 * * * /usr/bin/certbot renew --quiet` su crontab

Vigilancia

JHipster viene con soporte de monitoreo completo de Micrometer (https://micrometer.io/) .

En desarrollo, los datos de métricas estarán disponibles a través de JMX: inicie su JConsole y podrá acceder a ella.

En producción, su aplicación expone sus datos de métricas en un punto final que un servidor Prometheus (https://prometheus.io/docs/introduction/overview/) puede eliminar a intervalos regulares, dependiendo de lo que haya configurado.

(https://developer.okta.com/?utm_campaign=display_website_all_multiple_dev_dev_jhipster-q2_&utm_source=jhipster&utm_medium=cpc)

Construya arquitecturas de microservicios con JHipster y OAuth 2.0 (https://developer.okta.com/blog/2019/05/23/java-microservices-spring-cloud-config?utm_campaign=text_website_all_multiple_dev_dev_jhipster-q2_&utm_source=jhipster&utm_medium=cpc)

(http://www.octoconsulting.com/)

Octo Consulting Group (http://www.octoconsulting.com/)

- Equipo (https://www.jhipster.tech/team/)
- Ilustraciones (https://www.jhipster.tech/artwork/)
- Políticas (https://www.jhipster.tech/policies/)
- Meetups (https://www.jhipster.tech/meetups/)
- Archivo (https://www.jhipster.tech/documentation-archive/)

(https://github.com/jhipster/generator-jhipster)

(https://twitter.com/jhipster)

(https://gitter.im/jhipster/jhipster_hipster)