Title: Ensuring spring boot and liquibase receive and handle SIGTERM Post Body:

Currently running SpringBoot applications in a containerised environment (ECS) and I've observed scenarios in which the container gets terminated during start-up and while it's still holding the Liquibase changelock.

This leads to issues in all containers that are spun afterwards and ends up requiring manual intervention.

Is it possible to ensure that if the process receives a SIGTERM, it will gracefully handle termination and release the lock?

I've already ensured that the container is receiving the signals by enabling via InitProcessEnabled (in the CloudFormation template) and use of 'exec java ...' as a java agent we use does gracefully shutdown on this circumstances.

Accepted Answer:

Heyo,

As mentioned in the GitHub issue I have a workaround. A solution is yet to be implemented.

You can manually register a shutdown hook before running spring boot.. That hook should assure that the Termination is postponed until liquibase is done.

And the hook:

package dang; import lombok.SneakyThrows; import lombok.extern.slf4j.Slf4j; import java.util.Map; @Slf4j public class Grace

package dang; import org.springframework.boot.autoconfigure.SpringBootApplication; import org.springframework.boot.builder.SpringBootApplication;

EDIT

After implementing my workaround a contributor of liquibase shared a different solution (It's actually the same solution just through Spring functionality) which is much better than what I did:

package dang; import org.springframework.boot.autoconfigure.SpringBootApplication; import org.springframework.boot.builder.SpringBootApplication;

Highest Rated Answer: None