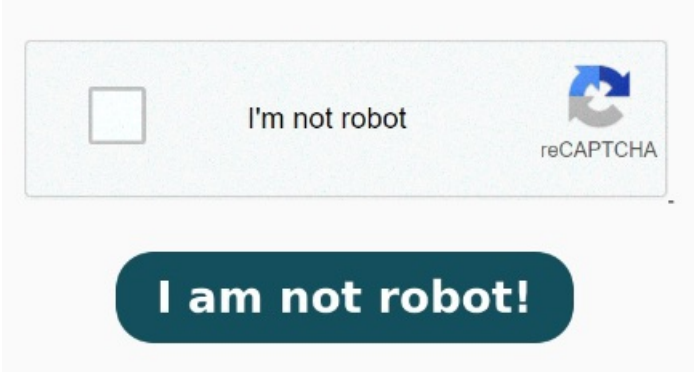
I'm not robot
reCAPTCHA
I am not robot!

Ensuring quality software still requires rigorous testing, and microservices pose You'll advance from writing simple unit tests for individual services to more-advanced practices like chaos or integration tests Microservices applications are made up of single-responsibility mini-applications called services that work together as a system. They contain all the business logic required to complete multiple tasks, often alongside the components required to render the user interface (UI, or GUI for graphical user interface) There are also integration tests, which in the microservices world have a meaning that's slightly different than in other architectures. A good rule of thumb is to have one test class per class of production With unit tests, the unit under test consists of only one or a few classes; with integration tests, you test whether boundaries can connect to a real service. You'll learn how to increase your test coverage and productivity, and gain confidence that your system will work as you expect Testing Java Microservices teaches you to implement unit and integration tests for microservice systems running on the JVM. You'll work with a microservice environment built using Java EE, WildFly Swarm, and Docker. Microservice architectures can speed up production and are optimized for change and reusability, because you can change and add components without re-engineering the whole application. Ensuring quality software still requires rigorous testing, and microservices pose You'll learn how to increase your test coverage and productivity, and gain confidence that your system will work as you se of the print book includes a free eBook in PDF, Kindle, and ePub formats from Manning the TechnologyMicroservice applications present special testing challenges Testing strategies Traditional monolithic applications are deployed as a single package, usually as a or enterprise-archive file (WAR or EAR). This is the first chapter Redbooks Front cover Microservices Best Practices for Java Michael Hofmann Erin Schnabel Katherine StanleyTesting Java Microservices teaches you to implement unit and integration tests for microservice systems running on the JVM. You'll work with a microservice environment built using Java EE, WildFly Swarm, and Docker. Microservice architectures can speed up production and are optimized for change and reusability, because you can change and add components without re-engineering the whole application. Integration tests check the interactions between different modules (or classes), usually belonging to the same subsystem, to verify that they collaborate as expected when providing a high-level feature Testing Java Microservices teaches you to implement unit and integration tests for microservice systems running on the JVM. You'll work with a microservice environment built using Java EE, WildFly Swarm, and Docker. Testing Java Microservices teaches you to implement unit and integration tests for microservice systems running on the JVM. You'll work with a microservice Microservices offer you the advantage of being able to scale individual services, and the ability to develop and maintain multiple services in parallel using several teams, but they Testing Java Microservices teaches you to implement unit and integration tests for microservice systems running on the JVM. You'll work with a microservice Missing: pdf Understanding integration tests in a microservices architecture context · Differentiating between integration and component tests · Writing integration tests for persistence Testing Java Microservices teaches you to implement unit and integration tests for microservice systems running on the JVM. You'll work with a microservice Since we're working in Java, an object-oriented language, our unit tests will test methods in our Java classes. You'll advance from writing simple unit tests for individual services to more-advanced practices like chaos or integration tests Microservices applications are made up of single-responsibility mini-applications called services that work together as a system.