Stateful vs stateless

Updated December 21, 2023

Copy URL

Overview

The state of an application (or anything else, really) is its condition or quality of being at a given moment in time—its state of being. Whether something is stateful or stateless depends on how long the state of interaction with it is being recorded and how that information needs to be stored.

8 considerations when adopting cloud-native apps

Stateful

Stateful applications and processes allow users to store, record, and return to already established information and processes over the internet. In stateful applications, the server keeps track of the state of each user session, and maintains information about the user's interactions and past requests. They can be returned to again and again, like online banking or email. They're performed with the context of previous transactions and the current transaction may be affected by what happened during previous transactions. For these reasons, stateful apps use the same servers each time they process a request from a user.

If a stateful transaction is interrupted, the context and history have been stored so you can more or less pick up where you left off. Stateful apps track things like window location, setting preferences, and recent activi
transactions as an ongoing periodic conversation

What brings you here today?

2

The majority of applications we use day to day are stateful, but as technology advances, microservices and containers make it easier to build and deploy applications in the cloud.

APIs help facilitate stateless apps →

Stateless

A stateless process or application, however, does not retain information about the user's previous interactions. There is no stored knowledge of or reference to past transactions. Each transaction is made as if from scratch for the first time. Stateless applications provide one service or function and use a content delivery network (CDN), web, or print servers to process these short-term requests.

An example of a stateless transaction would be doing a search online to answer a question you've thought of. You type your question into a search engine and hit enter. If your transaction is interrupted or closed accidentally, you just start a new one. Think of stateless transactions as a vending machine: a single request and a response.

Stateful vs stateless: a comparison

The key difference between stateful and stateless is whether an application retains information about the current state of a user's interactions or if it treats each request as an independent, isolated transaction. However there are also specific differences including:

- **Scalability**: Stateless applications are generally more scalable, as each request is independent and can be handled by any available server. Stateful applications may require more complex mechanisms for load balancing and session management.
- **Fault tolerance**: Stateless applications can be more fault-tolerant, as the loss of a server doesn't impact user sessions. In stateful applications, the loss of a server can result in the loss of session data unless additional measures, such as session replication or clustering, are in place.
- **Resource utilization**: Stateless applications often have lower resource utilization because there is no need to store and manage session data. Stateful

applications may require more memory and processing power to handle and maintain session information.

- **Development complexity**: Stateless applications can be simpler to develop and maintain, as there is no need to manage state across multiple requests. Stateful applications, on the other hand, require careful handling of session data and state management.

Containers and state

As cloud computing and microservices grow in popularity, so too has containerization of applications, whether stateful or stateless. Containers are units of code for an application that are packaged up, together with their libraries and dependencies, so that they’re able to be moved easily and can run in any environment, whether on a desktop, traditional IT infrastructure, or on a cloud.

Originally, containers were built to be stateless, as this suited their portable, flexible nature. But as containers have come into more widespread use, people began containerizing (redesigning and repackaging for the purposes of running from containers) existing stateful apps. This gave them the flexibility and speed of using containers, but with the storage and context of statefulness.

Because of this, stateful applications can look a lot like stateless ones and vice versa. For example, you might have an app that is stateless, requiring no long-term storage, but that allows the server to track requests originating from the same client by using cookies.

Stateless and stateful container management

With the growth in popularity of containers, companies began to provide ways to manage both stateless and stateful containers using data storage, Kubernetes, and StatefulSets. Statefulness is now a major part of container storage and the question has become not if to use stateful containers, but when.

Whether or not to use stateful or stateless containers comes down to a matter of what kind of app you’re building and what you need it to do. Stateless is the way to go if you just need information in a transitory manner, quickly and temporarily. If your app requires more memory of what happens from one session to the next, however, stateful might be the way to go.

Stateful, stateless, and Red Hat

When it comes to stateful or stateless, Red Hat has you covered. Whether you're orchestrating stateful containers on our enterprise-ready Kubernetes platform, Red Hat OpenShift, or creating a unified environment for app dev with Red Hat Integration, you'll be backed by our award-winning support and the industry's largest ecosystem of partners.

Red Hat OpenShift provides a unified, security-focused, hybrid cloud application platform that allows organizations to accelerate innovation by helping modernize their application development and deployment and operational processes. It also offers complete flexibility to organizations in where and how they run their application platform.

Red Hat Integration provides service composition and orchestration, application connectivity and data transformation, real-time message streaming, change data capture, and API management—all combined with a cloud-native platform and toolchain to support the full spectrum of modern application development.

See how all of our products build solutions, improve developer productivity, and promote innovation—the open source way.

Read about containerizing stateful apps in the cloud →

Keep reading

ARTICLE

Stateful vs stateless

Whether something is stateful or stateless depends on how long the state of interaction with it is being recorded and how that information needs to be stored.

Read more →

ARTICLE

What is Quarkus?

Quarkus is a Kubernetes-native Java stack made for Java virtual machines (JVMs) and native compilation, optimizing Java specifically for containers.

Read more →

ARTICLE

What is serverless?

Serverless is a cloud-native development model that allows developers to build and run applications without having to manage servers.

Read more →

More about cloud-native applications

Products   Related articles   Resources   Training

Red Hat
OpenShift

An enterprise application platform with a unified set of tested services for bringing apps to market on your choice of infrastructure.

Learn more →

About Red Hat

We're the world's leading provider of enterprise open source solutions—including Linux, cloud, container, and Kubernetes. We deliver hardened solutions that make it easier for enterprises to work across platforms and environments, from the core datacenter to the network edge.