Statements About Stateless

DevOpsDays Cairo 2024

Dan "phrawzty" Maher

Dan "phrawzty" Maher

- 👨‍💻 Open Source Engineer @ Cerbos
- 🧠 Co-Chair of DevOpsDays
- 💾 Previously Scaleway, Datadog, Mozilla, Ubisoft...
- 🛠️ A little more ops than dev ;)

cerbos

Cerbos

- Externalised, policy-based runtime authorisation for your applications.
- Open source, written in Go
- `https://cerbos.dev/`
- Most importantly: it's stateless!

Agenda

1. Core Principles
2. Advantages & Disadvantages
3. Practical Concerns

Agenda

0. What Even Is State?
1. Core Principles
2. Advantages & Disadvantages
3. Practical Concerns

What is state?

State refers to any information that a system or application needs to retain between different requests or interactions to understand and respond correctly to subsequent requests from the same user or process.

Examples of state

- User sessions
- Request context
- Client-specific data
- System state

Is stateless real?

Core principles

- Independent requests
- External state management
- Idempotency
- Decoupled components
- Horizontal scalability

Independent requests

- Every request is self-contained
- Every request is (considered) a fresh interaction

External state management

- State is managed outside of the interaction
- Client or external system manages continuity

Idempotency

- Same request, same result
- "referential transparency"

Decoupled components

- No shared state means forced modularity
- Components communicate through interfaces

Horizontal scalability

- Distributed workload by nature
- Cloud native (cloud-friendly?)

Advantages & Disadvantages

- Independent requests
- External state management
- Idempotency
- Decoupled components
- Horizontal scalability

Independent requests

- Advantages: Resilient, flexible, and distributed
- Disadvantages: Increased overhead, high network dependence

External state management

- Advantages: Simplified server-side, straightforward scalability
- Disadvantages: "Unusual" security profile, state synchronisation challenges

Idempotency

- Advantages: Improved reliability, graceful error recovery
- Disadvantages: Reduced flexibility, potentially complex implementation

Decoupled components

- Advantages: Modular, flexible, fault-tolerant
- Disadvantages: Coordination overhead, network intensive, sensitive to latency

Horizontal Scaling

- Advantages: Elasticity, load distribution, resilience
- Disadvantages: Now you're managing a distributed system. My condolences.

Practical Concerns

- Handling user sessions
- Caching mechanisms
- Deployment and lifecycle

Image courtesy Bent Inge Johansen (public domain): https://flic.kr/p/tjuZMw

Handling user sessions

- Externalised session management
- Client tokens (e.g. JWT)

JSON Web Tokens (JWT)

- Good: Stateless, compact, cryptographically secure signature
- Bad: Difficult to revoke, plaintext paylod

Caching mechanisms

- Distrbuted key/value store
- HTTP caching at the edge
- Browser cache

Test your caching mechanisms!

- Load testing, misses, unexpected invalidations, consistency concerns...

Deployment and lifecycle concerns

- Deployment / environment consistency
- Dependencies and service discovery
- Load balancing and traffic management

Deployment / environment consistency

- Side effects and emergent properties
- Situational differences

Dependencies and service discovery

- State management
- So many services!

Load balancing and traffic management

- Balancing algorithms and affinity
- (Auto) Scaling

Conclusion

- Everything is a trade-off

Actually the conclusion