

Stateful Architectures in Akka

Nick Isaacs
@jorge_distress
Sr. VP Engineering @

VictorOps



What Is This Talk?

- Anecdotal
- Informational
- A work in progress
- First steps



Why



Computers



VictorOps



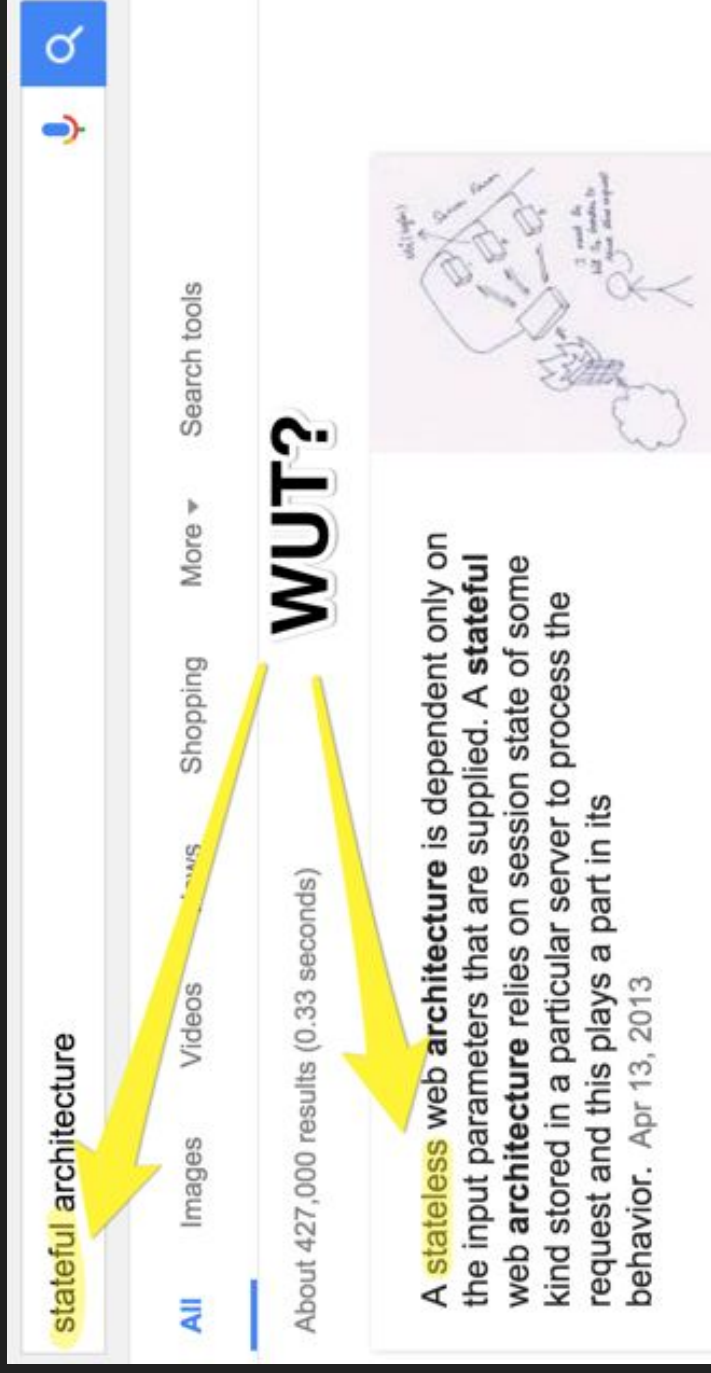
Users



Stateful Architecture



Stateful what?



The image is a screenshot of a Google search interface. The search bar at the top contains the text "stateful architecture". Below the search bar, the results are categorized by "All", "Images", "Videos", "Shopping", and "More". The search results show "About 427,000 results (0.33 seconds)". A large yellow arrow points from the word "stateful" in the search bar to a text box on the right. Another yellow arrow points from the word "WUT?" in the search results to the same text box. The text box contains the following text:

A stateless web architecture is dependent only on the input parameters that are supplied. A **stateful web architecture** relies on session state of some kind stored in a particular server to process the request and this plays a part in its behavior. Apr 13, 2013

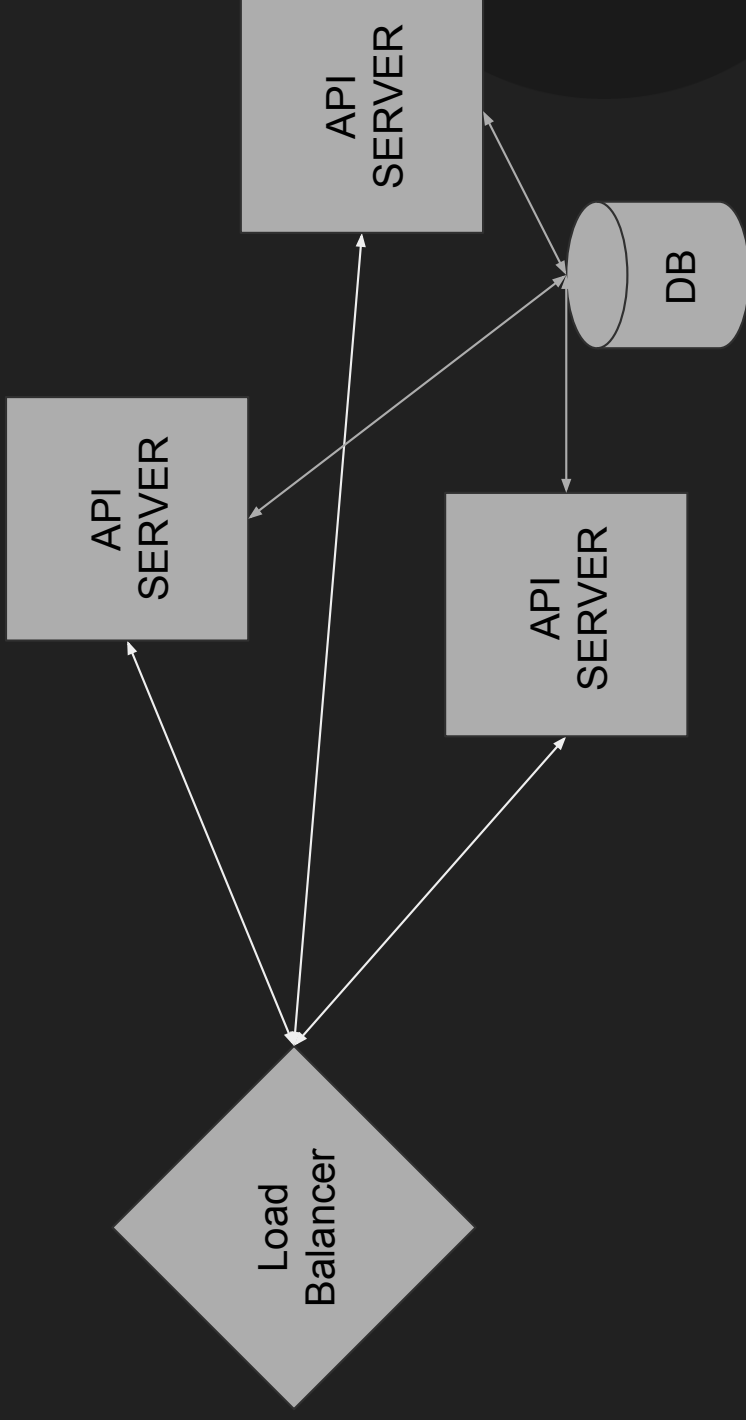
WUT?



The diagram shows a client (represented by a stick figure) interacting with a server. In the stateful architecture, the client sends a request to the server, and the server responds with a response. In the stateless architecture, the client sends a request to the server, and the server responds with a response. The diagram also shows a session state (represented by a cloud) that is shared between the client and the server.

Stateless Architectures

A typical load balanced API server cluster



Stateless Request Flow

1. Hit load balancer
2. Hit API box
3. Lookup state from persistence layer
4. ???
5. Respond
6. Profit



Stateless Request Flow

1. Hit load balancer
2. Hit API box
3. Lookup state from persistence layer
4. ???
5. Respond
6. Profit



Stateful Request Flow

1. Hit load balancer
2. Hit API box
3. ~~Lookup state from persistence layer~~
4. ???
5. Respond
6. Profit



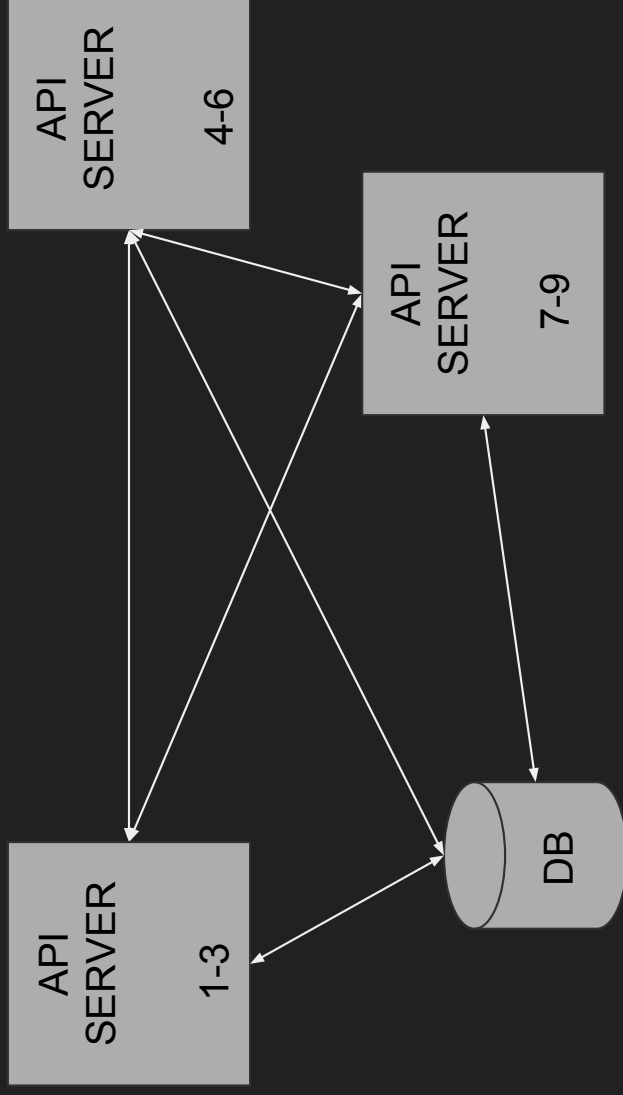
Strategies

- Ephemeral applications (avoid state)
- Caching
- Sharding



Stateful Architecture

A stateful API cluster. Each node has a subset of the application domain.



Sharding Applications

- Keeping data/logic in a known place
- Membership
- Consistent hashing
- Routing
- Coordination



What is Akka?

- Actor implementation on the JVM
- Actors send and receive messages
- Actors can “become” a new state between messages
- Actors are “islands of state”
- You cannot directly access an Actor’s state
- Many great intro talks/posts available



Akka to the Rescue

- Fantastic abstraction of state
- Persistent actors
- Rich message delivery semantics
- Simple message passing across the network
- Failure is a first class citizen



A Little Example

- TemperatureActor
- All of the state is managed in memory
- Fault tolerant, replayable log of our current state

<https://github.com/nicky-isaacs/SCALE-14x-demo>



1. Actor comes to life, time to read back the state

```
override def receiveRecover: Receive = {  
  case SnapshotOffer(_, TemperatureSnapshot(t)) =>  
    this.currentTemperature = Some(t)  
  
  case cmd : TemperatureCmd =>  
    updateState(cmd)  
}
```

2. Actor is recovered, time to work

```
override def receiveCommand: Receive = {  
  case GetTemperature =>  
    this.currentTemperature match {  
      case Some(t) =>  
        sender() ! t  
  
      case _ =>  
        sender ! AkkaFailure(new TemperatureException("No temperature to get"))  
    }  
  
  case cmd: TemperatureCmd =>  
    persist(cmd) { persistedCmd =>  
      if (shouldSnapshot()) {  
        this.currentTemperature foreach { t =>  
          snapshotAndSweep(TemperatureSnapshot(t))  
        }  
      }  
  
      updateState(persistedCmd) match {  
        case Success(_) =>  
          sender() ! AkkaSuccess(())  
  
        case Failure(err) =>  
          sender() ! AkkaFailure(err)  
      }  
    }  
}
```

Finite State Machine (thanks Akka)

- Recovering
 - No messages from the outside world
 - Reading snapshots and journaled messages
 - Getting back to last known state
- Receiving
 - Ready to respond to the world
 - State changes go to disk asynchronously
 - writes are all in memory, performance++



How We Cluster Today

- Stateful sharding of applications
- Zookeeper based membership
- Consistently hashed by organization
- Single leader



Where We Could Go

- Leaderless clusters
- Gossip membership protocol
- Replication of work as cluster scales



Extra Credit

- Uber Ringpop
 - “application-layer sharding for Node.js applications”
- Akka Clustering
- Dynamo paper



“I regret nothing. The end.”

- Ron Swanson

