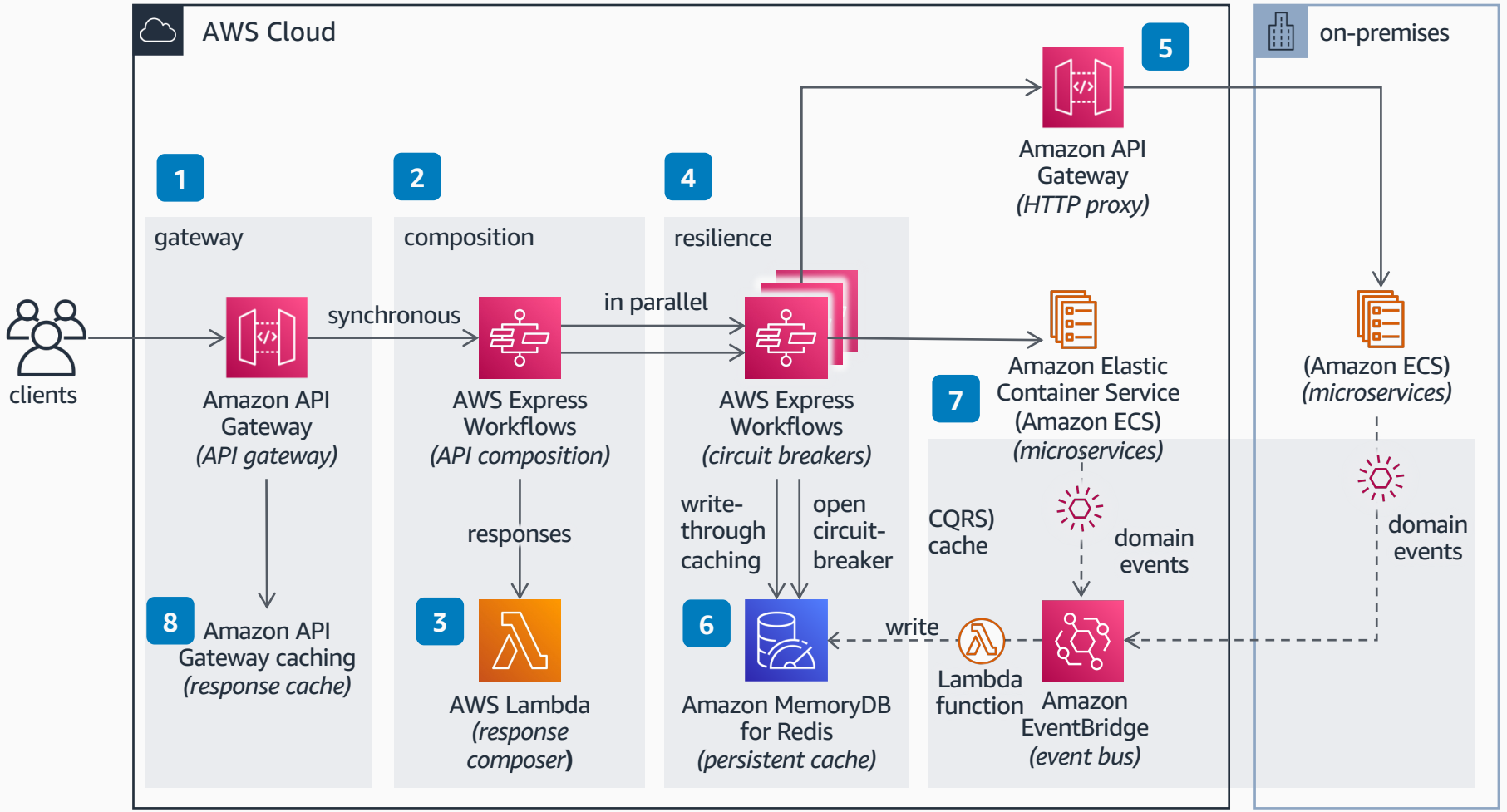
Parallel API Composition in AWS

Use this architecture if you need to call multiple downstream API endpoints, in parallel or in sequence, to compose a single aggregated response to your clients. Build a generic, parameterized circuit-breaker workflow, and build a custom workflow composition for each use case. Each composition calls multiple circuit breaker sub-workflows (one per downstream target). Increase availability of circuit breaker-related data using command query responsibility segregation (CQRS) to maintain a persistent eventually-consistent cache.



1. To allow clients to send synchronous web requests to your microservices, create an API gateway using **Amazon API Gateway**.
2. To invoke multiple microservices at the same time, use **AWS Express Workflows** to create a workflow with parallel steps.
3. Use an **AWS Lambda** function to compose the multiple responses received from the downstream microservices into a single aggregated response to return to clients.
4. To increase resilience, handle parallel requests with circuit breakers built with parameterized nested **AWS Step Functions** Express Workflows.
5. To invoke on-premises microservices, use **API Gateway** to proxy the HTTP requests.
6. Create an **Amazon MemoryDB for Redis** database to cache the responses from the microservices, using the write-through caching pattern. When a downstream microservice becomes unavailable or its latency reaches a threshold, the circuit is closed and all responses are read directly from the cache until the circuit reopens.
7. To increase availability, complement the cached data with the domains events raised by the downstream microservices using the CQRS pattern. Using **Amazon EventBridge**, create an event bus to collect the domain events, then handle them with a **Lambda** function to persist the domain aggregates in **MemoryDB**.
8. To increase performance, enable the **API Gateway** response cache, adjusting time-to-live (TTL) values and filters according to each request type.

Reviewed for technical accuracy May 17, 2022

AWS Reference Architecture