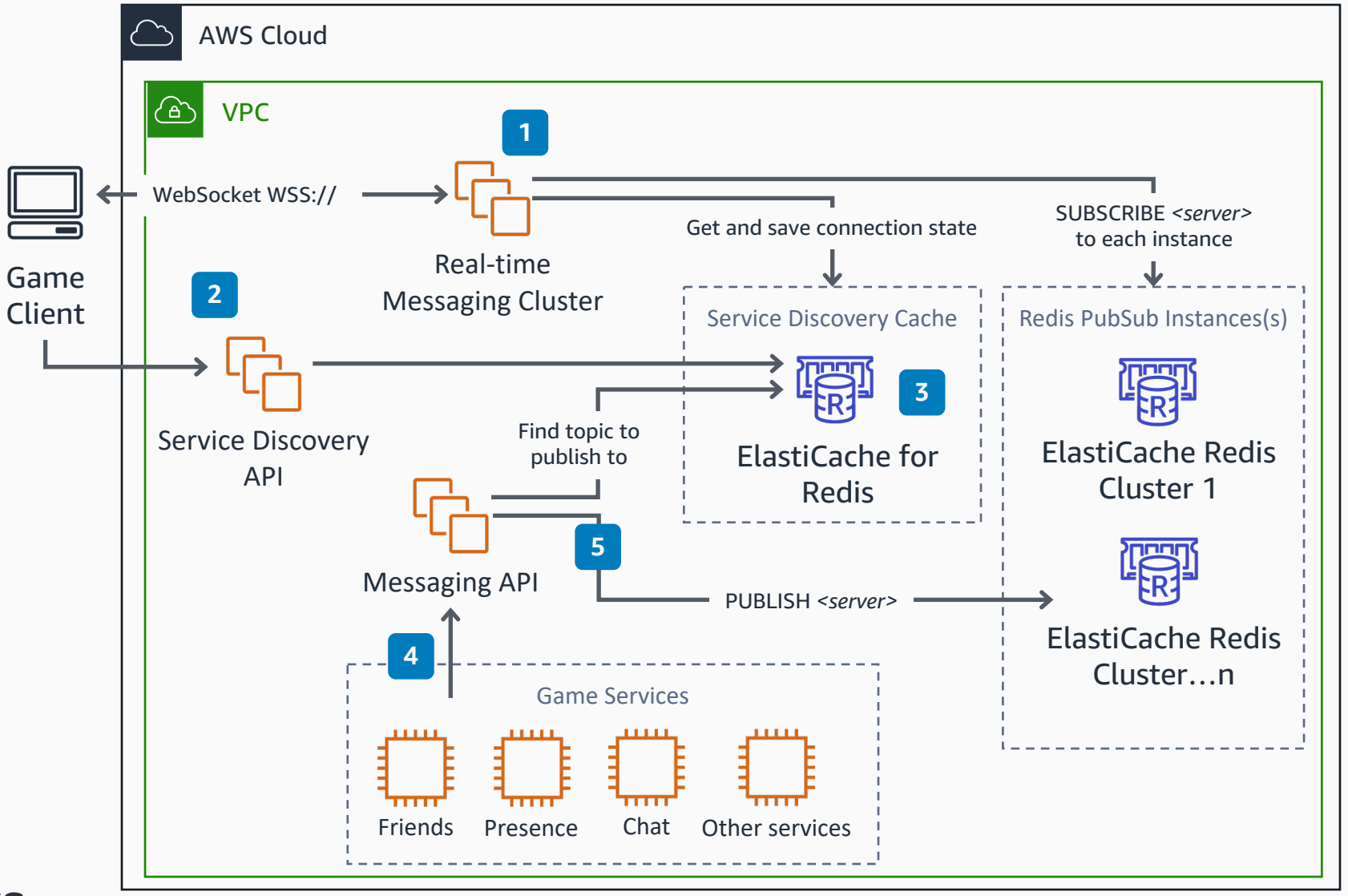
Massive Scale Real-Time Messaging for Multiplayer Games

Real-time messaging for delivering notifications to players in multiplayer games that can scale to support millions of concurrent users (CCU) using multiple Redis PubSub clusters with Amazon ElastiCache and WebSockets.



1. Multiple real-time messaging servers can be deployed into a cluster to provide horizontal scalability. When a real-time messaging server starts, it can check a Service Discovery cache using **Amazon ElastiCache for Redis** for a list of available Redis PubSub instances and subscribe to the same topic (i.e. real-time-messaging123) in each of the Redis PubSub instances. This enables traffic to be sharded across multiple ElastiCache clusters and still reach the correct real-time messaging server.

2. Clients can send requests to a Service Discovery API to retrieve a real-time messaging server to connect to. The list of available real-time messaging instances is stored in the Service Discovery cache.

3. A Service Discovery cache stores information about the connected endpoints, including clients, real-time messaging instances and the Redis PubSub clusters and topics that each real-time messaging instance is reachable on. If client disconnects, this information is removed from the cache.

4. Applications use a Messaging API to deliver messages to clients. The API publishes the message to the appropriate topic that reaches the correct real-time messaging server and game client.

5. The Messaging API publishes the message to the topic in one of Redis PubSub instances. Since each real-time messaging server is listening to the same topic in every Redis PubSub instance, it gets a message that is delivered to any of the ElastiCache clusters and delivers the message to the client, enabling horizontal scalability.

AWS Reference Architecture