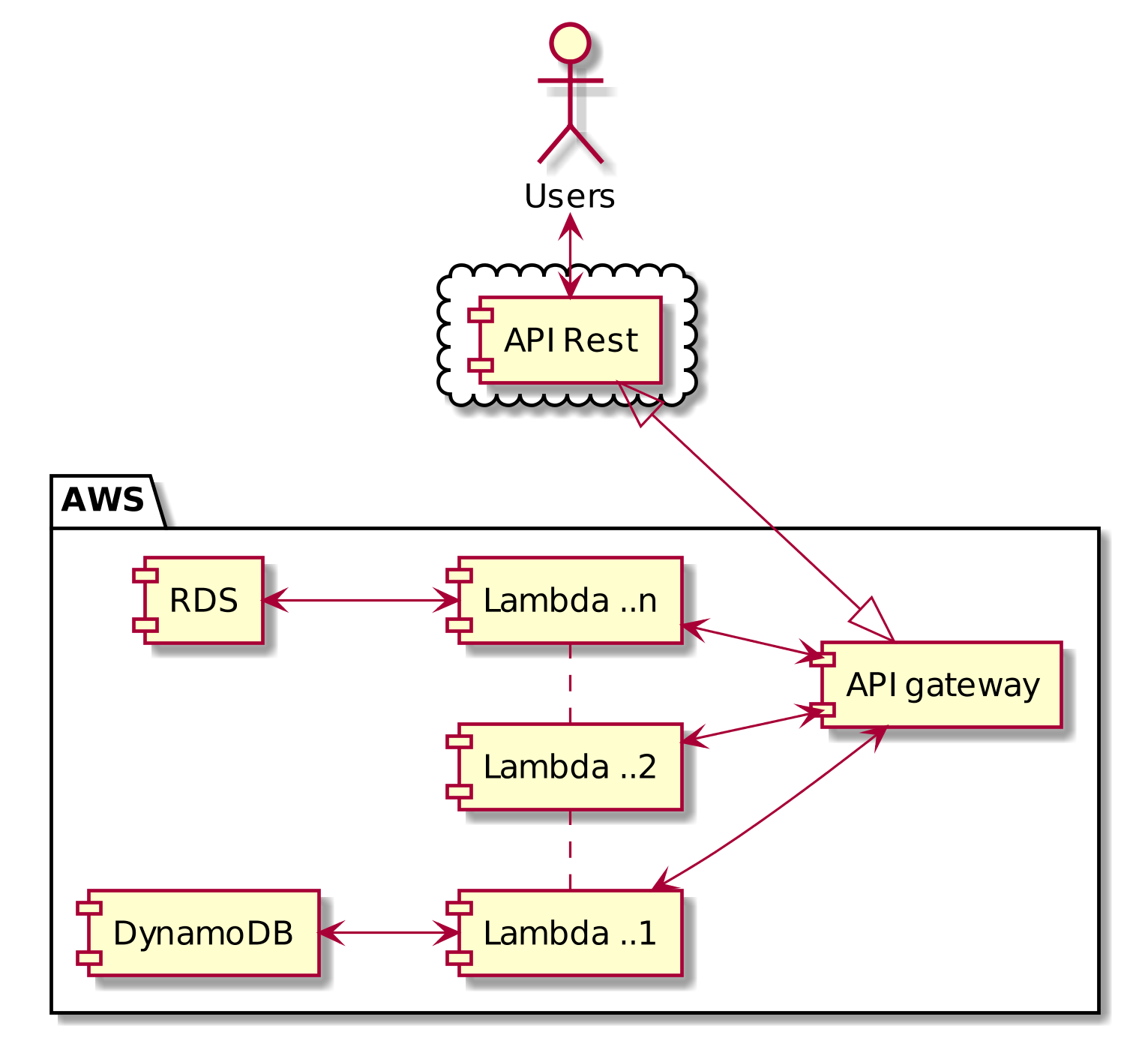
# 2. A restful solution



We have a Restful solution, here we have two kinds of data repositories, one is for Amazon Relational Database Service (RDS) for legacy data from a monolithic architecture and also we have an Amazon DynamoDB for NoSQL data were we will migrate from a relational DB into a NoSQL documents for an analytics process.

For the core data service we have some microservices were we put the business logic, one important point is that we have services related to authentication and authorization, security and also session management from a legacy system, but this services will be delegated to the AWS API gateway were the AWS API gateway has all this integrated services out of the box, in the lambda services in specific Lambda 2, here could be have an scheduler to do batch processing to migrate SQL into NoSQL, this service could be into a scheduler or in demand according complex domains rules. Into de “lambda n” service will act like a service cache to get the domain data ready to be exposed to a third client, this service would be implemented with a AWS ElastiCache to get better performance, but instead the domain logic requires some pre-processing will have implemented before to be a data cached.

The Amazon API Gateway is a fully managed service that makes it easy to create, publish, maintain, monitor, and secure APIs at any scale. This acts like a front door for applications to access data, business logic and functionality from the back-end services, such as workloads running on Amazon Elastic Compute Cloud (Amazon EC2), code running on AWS Lambda, or any web application also handles all the tasks involved in accepting and processing up to hundreds of thousands of concurrent API calls, including traffic management, authorization and access control, monitoring, and API version management, also we can use service like swagger to describe the API Gateway extensions to the Swagger specification. We need remember that Swagger aides in development across the entire API lifecycle, from design and documentation, to test and deployment.

This architecture is the core for many solutions that I have been implemented, also we can add a log filter to tracking all the activity into the hole system if the requirements are needed

And important part from develop is the capability to do performance testing in Continuous Delivery, most software delivery teams have practicing or planning to practice some flavor of continuous delivery. Its popularity has exploded in recent years largely because it has proven to have immense benefits for the rapid release of high-quality software. After each commit, the software is built and tested, and a deployable artifact is the result. How or when that artifact is deployed to either a staging area or to production depends on the team, their process, and their infrastructure.

So, the AWS CodePipeline is a continuous integration and continuous delivery service for fast and reliable application and infrastructure updates. CodePipeline builds, tests, and deploys your code every time there is a code change, based on the release process models that we define.