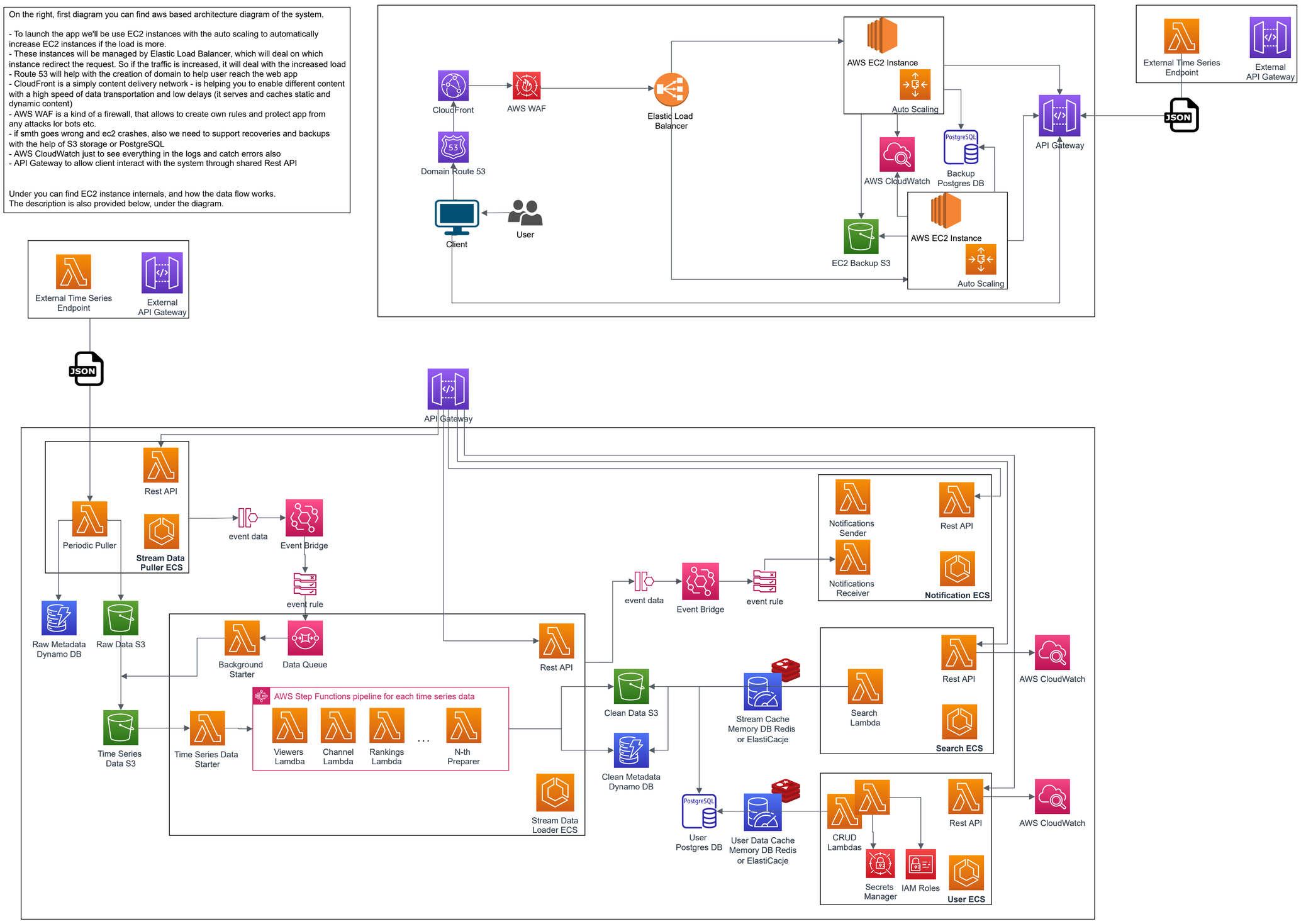
On the right, first diagram you can find aws based architecture diagram of the system.

- To launch the app we'll be use EC2 instances with the auto scaling to automatically increase EC2 instances if the load is more.
- These instances will be managed by Elastic Load Balancer, which will deal on which instance redirect the request. So if the traffic is increased, it will deal with the increased load
- Route 53 will help with the creation of domain to help user reach the web app
- CloudFront is a simply content delivery network - is helping you to enable different content with a high speed of data transportation and low delays (it serves and caches static and dynamic content)
- AWS WAF is a kind of a firewall, that allows to create own rules and protect app from any attacks lor bots etc.
- if smth goes wrong and ec2 crashes, also we need to support recoveries and backups with the help of S3 storage or PostgreSQL
- AWS CloudWatch just to see everything in the logs and catch errors also
- API Gateway to allow client interact with the system through shared Rest API

Under you can find EC2 instance internals, and how the data flow works.
The description is also provided below, under the diagram.



1. External request makes via external API endpoint
2. Stream Data Puller EC Service once per day / week / month (like a cron) makes requests to the external API to retrieve JSON time series data
3. Periodic puller stores this raw time series data in the Dynamo DB tables and S3 buckets (the data might be different, for example, some links, files will be stored in S3 buckets)
3.1. There is an option available to manually make requests through Rest API to retrieve some data from external API
4. When the data reaches storage, for each time series data event pushes via Event Bridge to the SQS and reaches then to the Background Starter lambda
5. Background Starter lambda then starts processing specific time series data (so it works for each data series, so there might be thousands of these time series): it copies data from the Raw Data S3 bucket to the Time Series Data S3 bucket. From now This Time Series Data S3 bucket - is the bucket which contains only one data series information, so in the next steps the pipeline will manipulate this piece of data.
6. When the data appears in the Time Series Data S3 bucket, it automatically triggers Time Series Data Starter lambda, which comes as a starting entry point to the Step functions pipeline to process and manipulate with series data by applying these lambdas inside SF
7. Each step of the SF manipulates the raw data depending on the functional requirements and needs. So each lambda (or inside there might be chains of lambdas per each functional requirement) do its processing of the data and as a result, provides the expected for the specific functional requirements results and store them in the Clean Data S3 bucket and Clean Metadata Dynamo DB.
8. Then the pipeline work finishes with the notification via Event Bridge (which might be switched off) for the client that results for this specific data series is ready to collect.
8.1. There is an option available to manually make requests through Rest API to run a pipeline for specific data series

9*. Each step is supported by Cloud Watch as a service to collect logs etc
10. Search EC Service is responsible for searching for the data that the user requests through the web app. This works via Rest API endpoints written for these needs, like serving information about concurrent viewers of each channel, games, ratings, etc. Data provided via these requests get from Clean storage. Interaction of the user with the functionality goes through Rest API endpoints.
10.1. For more fast access to the data there is an additional layer - Cache Memory DB (Redis).
11. User EC Service responsible for user data inside web app, their roles and security groups, what allowed for them to run or not, etc.

 # Viewers Lambda - extracts information from time series data on simultaneously watching videos for this time period, saving it to the database

 # Channel Lambda - pulls out information about the channel of the current stream for this time period from time series data, saving it to the database

 # Rankings Lambda - extracts information about channels, games, views from time series data, composing stream ratings for this data based on these data time interval, saving to the database

 # N-th Lambda - extracts information from time series data and does something with it, saving it to the database

 # Time Series Data S3 - background starter copies the data of a specific one time slot for 1 minute from Raw Data s3 to this bucket, after which when data enters the time series data s3 bucket, the launch of Time Series Data Preparer Lambda is triggered

 # Time Series Data Preparer Lambda - a function that extracts data from raw data that the end user needs for analytics, etc. Aggregating into certain sections, saving in the database. Views, all channels, channel and information about it, user activity, etc.

Cloud-native deployment

- Write a serverless.yaml file, where you can define the environment, stages, AWS services, etc
- Infrastructure as a code then goes on via usage of Cloud Formation. The files for it also need to be written as a yaml files.
- Also need to write builds. This code might be written via usage of .sh files
- CI \ CD pipelines.

- Also there might be used Terraform

CloudFormation