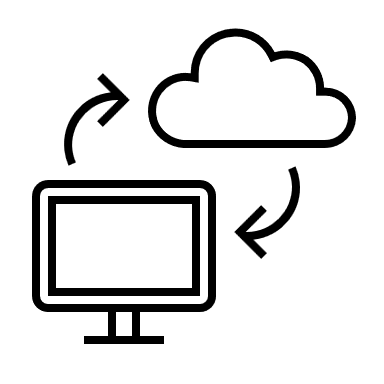
High-level design document

Cloud recommendations



Ashok Das & Tinku Manivikesh Chukkapalli

Table of Contents

[**1.** **introduction** 2](#_Toc116915422)

[**1.1.** **Purpose:** 2](#_Toc116915423)

[**1.2.** **Scope:** 2](#_Toc116915424)

[**1.3.** **Document Overview:** 3](#_Toc116915425)

[**2.** **recommendation categories** 4](#_Toc116915426)

[**2.1.** **Recommendation APIs in AWS:** 4](#_Toc116915427)

[**2.2.** **Recommendation APIs in GCP:** 5](#_Toc116915428)

[**3.** **System High-level design** 7](#_Toc116915429)

[**4.** **functionality** 8](#_Toc116915430)

[**5.** **built with** 9](#_Toc116915431)

[**6.** **Deployment** 9](#_Toc116915432)

[**Appendix** 11](#_Toc116915433)

[**1.** **values generated** 11](#_Toc116915434)

# **introduction**

## **Purpose:**

Now a days most of the organizations are moving to cloud for their operations which lead them to a major business challenge related to cost perspective. The cloud resources which are in non-production environments like Dev, QA, UAT, and Pre-Prod are underutilized and will be long-lasting ideally. Here every cloud provider will generate recommendations for the resources which are under-utilized or idle for a long period. Those recommendations can be applied to reduce cloud operational costs. But we need to apply it manually for each service or VM which will be more difficult for a large amount of utilization. So, we as a team came up with a solution called cloud recommendations in an automated way. Here the recommendations will be applied automatically in a scheduled time period using a cron job.

The main purpose of the document is to explain how to use recommendations in a cloud in an automated way. By applying these cloud recommendations to the resources, the organization’s cloud operational costs will be reduced by 25%-50%.

## **Scope:**

The scope of the document is to explain the process and steps involved in this process.

* Generating the recommendations using recommendations APIs
* After generating the recommendations are stored in a repository and the pull request is created for reviewing the recommendations.
* After the pull request is approved and merged the changes the recommendations apply pipeline job will be triggered automatically.
* Finally, the recommendations will be applied to the cloud resources as mentioned.

The cloud recommendations can be applied in an automated way by using the following:

* Jenkins
* Shell Script

**Jenkins:** Jenkins is a CI/CD tool here we will configure two Pipeline Jobs for applying recommendations:

1. **Recommendations Pipeline:** In this pipeline job by using recommendation APIs, we will get the recommendations from the cloud providers and store them in a file in a JSON format, and creates a pull request for approval to particular review admins. Once the review admin approves the second pipeline will automatically get triggered using webhooks. The recommendation JSON files areused to extract the recommended values.
2. **Recommendations Apply Pipeline:** This Pipeline will get triggered automatically once the review admin approves the changes and merge requests using webhooks. As soon as the pipeline got succeeded the recommendations will get applied to the cloud resources in the particular cloud provider.

**Shell Script:** Mainly for applying these recommendations we used shell scripting as a scripting language and also we used jQuery to extract the values from the recommendation file. Shell Script played a key role in applying the recommendations in an automated way.

## **Document Overview:**

There are four sections in this document that clearly explain the flow and functionality of cloud recommendations:

1. **Introduction:** A brief explanation of the purposes, goals, and format of this System Design Document.
2. **High-Level Design:** An overview of the goals and objectives for the Cloud recommendations. This section also provides a short explanation of each component and process to be developed/implemented.
3. **Functionality Explanations:** This section documents the detailed design of all Modules within the System.
4. **Deployment:** This section lists the options for deploying the system, and technical infrastructure where applications or processes should be set up in order to support the system.

# **recommendation categories**

The main aim of our team is to achieve the cost-optimization in the cloud. So we mainly concentrated on the recommendations which are related to cost-optimization as per our end goal. The following are the categorized recommendation APIs for the AWS and GCP related to the cost perspective.

## **Recommendation APIs in AWS:**

In AWS Cloud the recommendation APIs related to cost perspective are mainly categorized into two ways:

* AWS Cost explorer
* AWS Compute-Optimizer

**AWS Cost explorer:** In this section, there is one recommendation API that is related to computing-based recommendations related to cost optimization.

1. **RightSizing Recommendations:** Rightsizing Recommendations API is used to get the recommendations for idle or unused EC2 instances. The recommendations provided by the AWS Service are called AWS Cost-Explorer. And also create recommendations that help you save costs by identifying idle and underutilized Amazon EC2 instances. Recommendations are generated to either downsize or terminate instances, along with providing savings detail and metrics. By using the Rightsizing Recommendation API the estimated savings for the cloud costs will be reduced by 50%.

**AWS Compute-Optimizer:** In this section, there are four recommended APIs that are related to the compute-based recommendations related to cost optimization.

1. **EC2 Instance recommendations:** EC2 Instance Recommendations API is used to get the recommendations for idle or unused EC2 instances. The recommendations are provided by the AWS Service called Compute-Optimizer. And returns Amazon EC2 instance recommendations. AWS Compute Optimizer generates recommendations for Amazon Elastic Compute Cloud (Amazon EC2) instances that meet a specific set of requirements. By using the EC2 Instance Recommendation API the estimated savings for the cloud costs will be reduced by 25%.
2. **EBS Volume recommendations:** EBS Volume Recommendations API is used to get the recommendations for idle or unused EBS Volumes. The recommendations are provided by the AWS Service called Compute-Optimizer. And also returns Amazon Elastic Block Store (Amazon EBS) volume recommendations. AWS Compute Optimizer generates recommendations for Amazon EBS volumes that meet a specific set of requirements. By using the EBS Volume Recommendation API the estimated savings for the cloud costs will be reduced by 25%.
3. **Autoscaling Group recommendations:** Autoscaling Group Recommendations API is used to get the recommendations for idle or unused Autoscaling Groups. The recommendations are provided by the AWS Service called Compute-Optimizer. And returns Auto Scaling group recommendations. AWS Compute Optimizer generates recommendations for Amazon EC2 Auto Scaling groups that meet a specific set of requirements. By using the EC2 Autoscaling Group Recommendation API the estimated savings for the cloud costs will be reduced by 25%.
4. **Lambda Functions recommendations:** Lambda Function Recommendations API is used to get the recommendations for idle or unused Lambda Functions. The recommendations are provided by the AWS Service called Compute-Optimizer. And returns AWS Lambda function recommendations. AWS Compute Optimizer generates recommendations for functions that meet a specific set of requirements. By using the Lambda Function Recommendation API the estimated savings for the cloud costs will be reduced by 25%.

## **Recommendation APIs in GCP:**

In GCP Cloud the recommendation APIs related to cost perspective are mainly categorized into the following:

1. **Idle persistent disk recommender**: Compute Engine provides recommendations to help you identify resources like persistent disks (PDs) that aren't used. You can use an [idle persistent disk recommender](https://cloud.google.com/compute/docs/viewing-and-applying-idle-resources-recommendations) to help minimize the waste of resources and reduce your compute bill. For PDs that are not actively used, you can create a backup snapshot and then delete the resource. By taking a backup and deleting the PD it Reduces the maintenance cost of that disk by 35% to 92% and Saves00% of the cost of that disk.
2. **Idle custom image recommender:** Compute Engine provides recommendations to help you identify resources like custom disk images that aren't used. You can use idle custom image recommender: to help minimize waste of resources and reduce your compute bill. For images, you can delete them if you don't need them and Save 100% of the cost of that image.
3. **Idle IP address recommender:** Compute Engine provides recommendations to help you identify resources like IP addresses that aren't used. You can use an idle IP address recommender to help minimize the waste of resources and reduce your compute bill. For PDs that are not actively used, you can create a backup snapshot and then delete the resource. For unused PDs, images, and IP addresses, you can delete them if you don't need them. Save 100% of the cost of that IP address.
4. **Idle VM recommendations:** Compute Engine provides idle VM recommendations to help you identify virtual machine (VM) instances that have not been used. These recommendations are generated automatically based on system metrics gathered by the Cloud Monitoring service over the previous 14 days. You can use idle VM recommendations to find and stop idle VM instances to reduce the waste of resources and reduce your compute bill.
5. **VM machine type recommender:** Compute Engine provides machine type recommendations to help you optimize the resource utilization of your virtual machine (VM) instances. These recommendations are generated automatically based on system metrics gathered by the Cloud Monitoring service over the previous 8 days. Use these recommendations to resize your instance's machine type to use the instance’s resources more efficiently. This feature is also known as rightsizing recommendations.

# **System High-level design**

We as a team at Criticalriver working as a Digital Innovation Centre team building a solution called Cloud recommendations which will reduce the organization’s cloud operational costs by implementing this cloud recommendations solution. Here the recommendations are applied to the cloud resources automatically once the review admin approves without any manual effort.

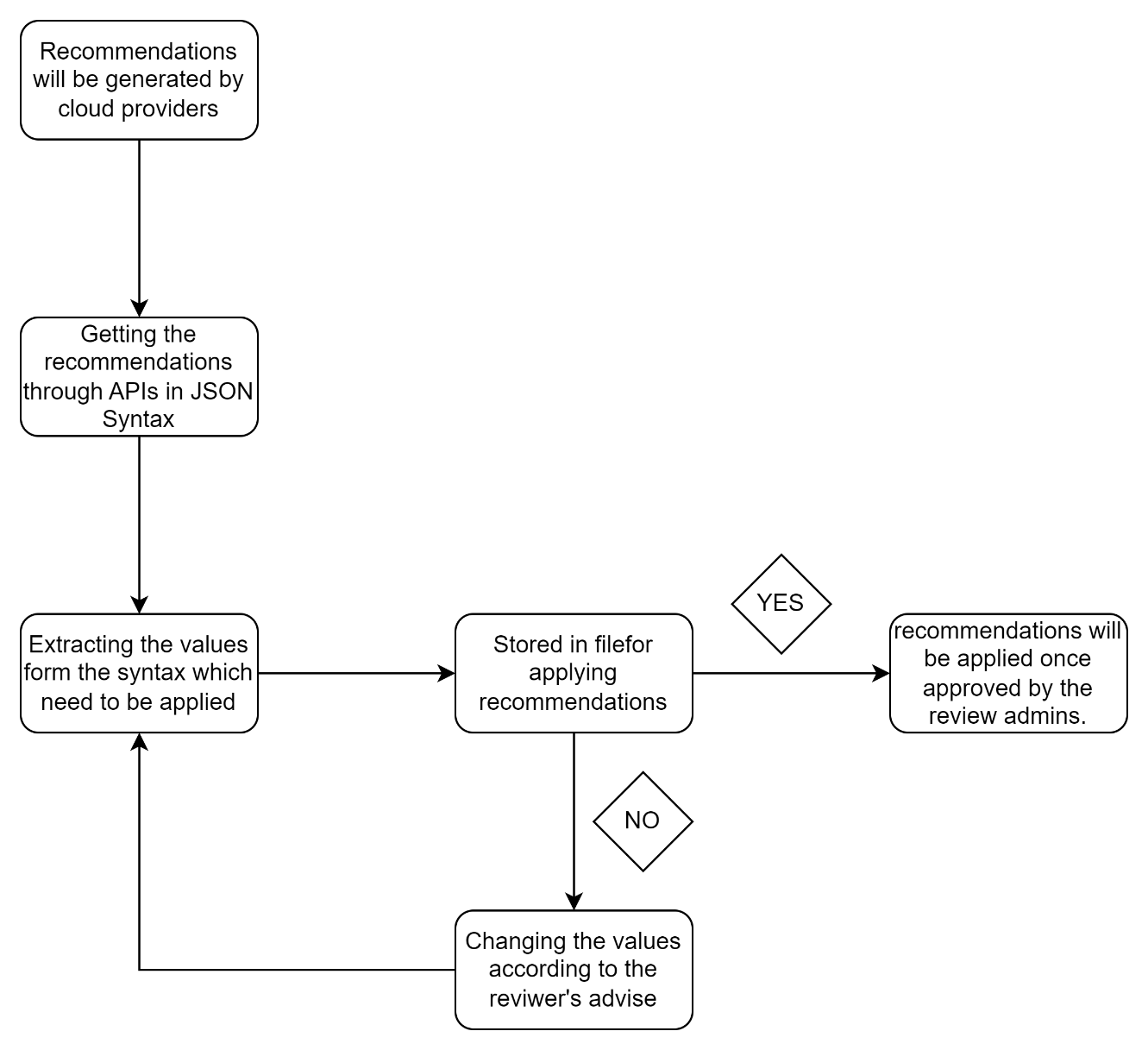
The following Design is the High-Level flow of explaining the actual function of the cloud recommendations

Diagram, schematic

Description automatically generated

# **functionality**

The overall function of the cloud recommendations is getting the recommendations by using APIs and extracting the recommended values shown in the JSON file. Those values are going to be applied once the reviewer approves the request for changing the recommendations. If the reviewer does not approve the request, we need to change the configuration as per the reviewer’s advice and the values also need to be changed because this process should not affect the system’s configuration which we are also taking into consideration like applying these recommendations should not affect the running applications so as per reviewer’s advice the values need to be changed and repeat the process. The overview of the process flow is given below.



# **built with**

* GitHub
* Jenkins
* Shell Script

# **Deployment**

The deployment of the cloud recommendations will take place in four different stages

1. **Getting recommendations using APIs:** In this step, the recommendations are generated using recommendations APIs. The generated recommendations are stored in a JSON format.
2. **Extracting values from the recommendations:** In this step, the values are extracted from the generated recommendations by running jQuery and stored it in a file for applying the recommendations.
3. **Creating Pull request:** In this step, after the values are extracted the pull request will be generated for review by the particular review administrator. After the review admin approves the pull request and merges the changes the apply pipeline will automatically get triggered using the webhooks.
4. **Applying recommendations:** Finally, the recommendations will be applied to cloud resources as mentioned values.
5. **Ignoring a few Changes:** In case the review admin doesn’t want to apply a few recommendations in the list he will comment in the pull request chat and the user will note the details and modify those recommendations list as advised by the review administrator and repeat the process flow.

**Deployment Overview:** The cloud recommendations deployment can be done by executing two pipeline jobs that are configured in Jenkins. One is for getting recommendations using APIs from cloud resources and the other is for applying recommendations to the cloud resources after approval by the review administrator. The two pipelines in Jenkins are given below.

1. **Recommendations pipeline:** In this pipeline, the recommendations are brought using the recommendation APIs by executing shell commands in respective cloud providers. The generated values for the recommendations are related to the configuration of the system and also the suggested recommended values. After getting the values it is stored in a file and will change the permissions as mentioned in the script. Later it will create a pull request and the notification will be sent to the review administrator.
2. **Recommendation apply pipeline:** This Pipeline will start automatically once the pull request has been approved and the changes have been merged with the help of preconfigured webhooks. As soon as the pipeline job got succeeded the recommendations will be applied to the particular cloud resources as mentioned. And the configuration will be changed or the resources will be deleted as mentioned in the recommendation values.

# **Appendix**

# **values generated**

The values generated by the recommendation APIs will get the output in JSON Syntax and the covered values include the following,

* Account ID
* Resource ID/Resource ARN
* Current Configuration
* Recommended Configuration
* Performance factor
* Risk Factor