CLOUD CAPACITY MANAGEMENT

Ashok Das & Tinku Manivikesh Chukkapalli

CRITICALRIVER

WHAT IS CAPACITY MANAGEMENT

In today's IT organization, it's not a question of whether public cloud resources are used for production applications, but which application should be deployed there. For modern digital businesses, capacity and cost management are essential to ensure adequate resources and budget, whether on-premises or in the cloud, to support new, existing, and growing business services. Nowadays most of the organizations from small scale to large scale are switching their 80% of their operations to the cloud. According to Gartner, approximately 28 percent of server capacity currently goes unused, as well as 40 percent of storage.

Capacity management informs forecasting and planning by helping you determine the capacity levels that you'll notice across your environment, including compute configurations, storage, database, and network bandwidth, as well as the most cost-effective way to provision them.

With the rise of easy-to-deploy and cheaper distributed servers and virtual machines, many organizations allowed capacity management to lapse, or moved to a performance management approach where performance issues are used to flag capacity issues. Of course, this also meant accepting the high cost of inefficient provisioning, as VMs proliferated throughout the environment without a clear understanding of the capacity or utilization of each server. Over time, as this sprawl bloated capital expenses, many organizations returned to capacity management at some level, whether using a formal tool or informal spreadsheets, notes, and approximations.

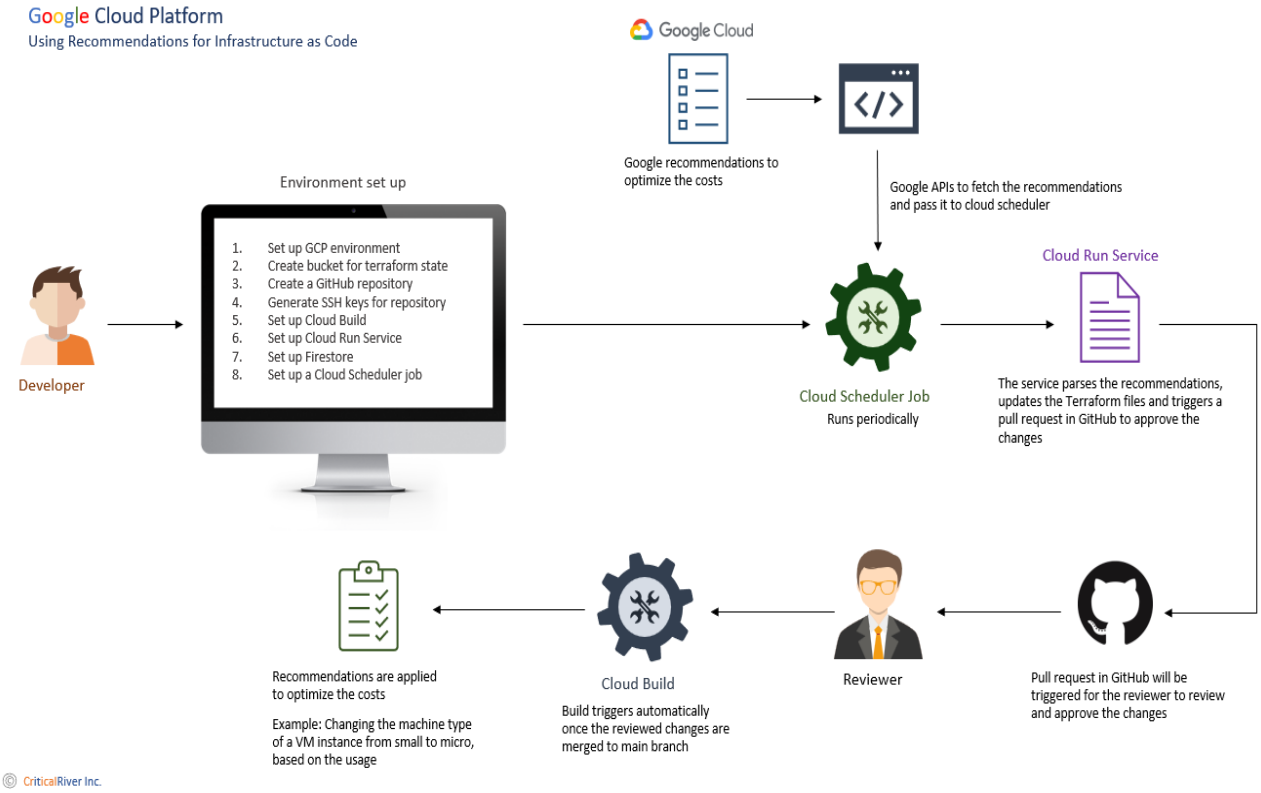
INSPIRATION FOR THE IDEA

One of the biggest overlooked factors driving up cloud costs is the number of services. Because of the elasticity of the cloud, teams can easily subscribe to the number of resources needed at a given time. Often resources are over-provisioned to meet the performance requirements of applications, resulting in the business paying higher costs than needed. Research from Gartner shows that without cloud cost optimization, business could be overspending by more than 70%. Without the right tool, organizations are going to drive up cloud costs. We as a team noticed that the organizations are completely reliant on a cloud for their 24/7 needs. So, to overcome this cost-effective challenge every cloud provider itself provides recommendations for idle and non-used services. Our main approach and achievement are to apply those recommendations in an automated way using the Continuous Integration (CI) tool through API Recommenders provided by the cloud provider.

OUR APPROACH

Mainly our aim is to apply the recommendations provided by the cloud in an automated way using the Continuous integration (CI) tool through API recommenders for services provided by the cloud provider. Firstly, we will prepare all the required configuration files like Infrastructure as a Code (IaaC), and JSON file format with the required syntax. The request syntax can be only in JSON format. After changing or updating any recommendations to the services by making changes to the code in GitHub. Once the changes or updates have been made the pull request is raised for review and a notification will be sent to the respective review administrator for reviewing the pull request. Once the pull request is approved the changes should be merged and the Infrastructure as a Code (IaaC) Configuration will automatically start the Continuous Integration (CI) Pipeline and the respective changes for the service will be applied automatically.

Framework



HOW ARE WE BUILDING IT

Our initial approach is to reduce cloud costs for organizations by providing recommendations through APIs in an automated way using the Continuous Integration (CI) tool. Firstly, the development will be done by setting up respective environments in the cloud. Later we will schedule the jobs using Cron Job to run periodically on desired timings in the cloud scheduler. Cloud Scheduler job runs by fetching recommendations from the recommendation list through recommender APIs provided by the cloud providers. After starting the scheduled job through the cloud scheduler job, the cloud run service will parse the recommendations and update the terraform files and triggers a pull request on GitHub to approve the changes and a notification will be sent to the respective review administrator. Later the cloud build will trigger the build automatically once the reviewed changes are merged to the main branch. Finally, respective recommendations are applied to optimize the cost.