ORACLE

Oracle Technology Engineering Specialists Best Practices

FinOps and Operations

7 March 2024 | Version 1.0

CONTENTS

AUTHORS

Name	Email	Role	Company
Andrew Bond		CTO and Specialist Leader	Oracle
Manuela Fioramonti		Manageability Specialist Leader	Oracle

HISTORY

Version	Authors	Date	Comments
1.0	Manuela Fioramonti	1st March 2024	Created a first version, with 'Operations Best Practices while managing a shared OCI Tenancy'

OPERATIONS BEST PRACTICES WHILE MANAGING A SHARED OCI TENANCY

This chapter has the scope of presenting the best practices the Oracle Technology Engineering team adopts while managing a shared environment for testing purposes. It encompasses the operations world, from FinOps to SecOps. For tenancies meant to include a production scope, we highly recommend considering and putting at work also D&R best practices, observability best practices and fleet management considerations for Compute, DB, Java and custom applications as well as a proven **Incident Management Procedure**.

FINOPS

Applying FinOps principles to an internal, testing tenancy is no different than applying them to a production tenancy. You need to move from the **Inform** phase to the **Optimize**, and finally to the **Operate** phase.

OCI provides tools to ensure FinOps discipline can be applied:

Organizations: allow a parent-child relationship between various tenancies. The Parent tenancy can centrally manage costs and governance.

Tenancies: provide a strong isolation among workloads.

Compartments: allow resources to be organized.

Resources: all OCI artifacts that can be deployed.

Tags: add metadata information to resources to apply governance policies and break down cost reporting.

Budgets: thresholds that can be set on costs per billing cycle. OCI will evaluate every hour to send an alert in case the value for the budget is crossed. Budgets can be targeted toward subscriptions, child tenancies, compartments, and tags. The integration with the Event service can be used to trigger automations such as the creation of Quotas.

Quotas: limits are set by admins on compartments to prevent overspending.

In a shared environment, especially if used for internal testing and development purposes, the challenge of keeping costs under control can become daunting. As team members come and go, team structures change, spun-up resources for testing may be forgotten, leading to unnecessary consumption and with it the risk of approaching resources limits, preventing team members from being able to complete their testing.

Cost reporting tools in OCI are aware of resource hierarchy and tags.

4.1 Three Steps to Plan a Cost Structure

Inform: Understanding your organization's reporting needs is the first step that needs to be taken into account when planning for a costing structure. In our case, every team needed to be able to see its consumptions, as an initial step. For this reason, the compartment structure was organized by creating a compartment for each team manager. Each individual in the team would have his/her compartment as a subcompartment to create his/her resources. Shared resources within the team would be created in the parent compartment.

During this phase, each team gets a clear understanding of their usage, their costs and their consumption details.

Cost Analysis reports are used to visualize consumption over time. Group-by and filtering dimensions are available via appropriate policies set-up. Queries can be saved for reuse. Data can be extracted as a CSV and as a PDF file. Future cost prediction is estimated based on current usage patterns. Reports can run at a specific time and deliver the content to an object storage. In addition, we are using **Custom Cost Reports** to show usage hourly per resource. Combined reports can be generated for all tenancies in an organization, viewed in Oracle Analytics Cloud, and queried in Autonomous DB when transferred. This report can be automated.

Optimize: This is the second phase of the FinOps framework. During this phase, we decided to take a two-sided approach:

From a pure cost show-back perspective, we automated custom reports and budget alerts. We decided not to implement quotas to prevent team members from being stopped in their work, however setting quotas is a recommended practice.

From a resource management perspective, we reach out to 'over spenders' to take corrective actions.

Note: In our case, there is no need for reporting cost and usage across multiple cloud providers. However, should this be one of the requirements, depending on the CSPs involved, it can be met by **3rd Party tools** such as Flexera, CloudHealth, and Cloudvane. Moreover, FinOps Open Cost and Usage Specification (FOCUS) is defining a common format for billing data, that empowers organizations to better understand cost and usage patterns across CSPs. The converters are being shared on Git by all contributors. Converters generate a normalized Parquet file for each CSP in a CSP-agnostic way. These files that can be imported in Autonomous DB.

Operate: This is the third phase of the FinOps framework. In this phase, OCI offers powerful tools to identify areas for improvement. The **Cloud Advisor** enables cost-saving opportunities with resource-specific recommendations and customizable profiles. These recommendations can be immediately implemented via the 'fix-it' flow, without navigating to the specific resources.

We realized that due to a lack of resource termination, no longer needed resources were kept running. As a side effect, Service Limits were encountered in creating new dynamic groups for example. To tackle this problem, we created scripts capable of identifying those resources across compartments, and implemented the following procedure:

1. owners are identified;
2. owners are notified to provide clarification;
3. if no answer is received within a certain timeframe, those resources are flagged for termination. Termination is handled via OCI-SuperDelete, available on GitHub. Otherwise, actions are taken accordingly to the clarification received.

4.2 SecOps

From a pure SecOps perspective, in the shared tenancy, the challenges we face every day are:

1. Identity Lifecycle Management: managing and retiring users and their roles and permissions.
2. Access Governance: ensuring access only to needed resources.

When embracing Infrastructure-as-Code, security needs to be accounted for, encompassing protection in CI/CD pipeline components, such as permissions, personal access tokens, vulnerability scanning on OCIR - OCI Registry, and approval workflows to ensure only approved changes, from allowed users, are deployed. This is what is called DevSecOps.

Just like for FinOps, OCI provides a set of powerful tools to help manage security operations over time:

Cloud Guard is a cloud-native service that helps you monitor, identify, achieve, and maintain a strong security posture in the Oracle Cloud. Also in this case, a clear design is needed before implementing it, such as mapping targets to compartments, the reporting region, and integration with other processes once a problem is detected.

Security Zones help prevent misconfigurations.

OCI Vulnerability Scanning checks hosts for potential vulnerabilities.

OCI Logging provides critical diagnostic information that describes how resources are performing and being accessed. Also in this case design decisions are key, such as the archiving period and SIEM integration.

Oracle Access Governance provides an IGA (Identity Governance and Administration) solution to help customers keep at bay access concerns and much more.