

# **XC3 User Manual - Project Breakdown**

**June 2023**

## **Technology Application Project**

Swinburne University of Technology

**Supervisor:** Dr. Ayesha Binte Ashfaq

**Team:** Nidhi Gupta  
Sahil Pruthi  
Jasmine  
Sijmen de Bruijn

## Table of Contents

1. Introduction	3
2. User Requirements	3
3. System Requirements and Installation Instructions	3
4. System Components	3
4.1 Architecture Overview	3
4.2 Lambda Function	4
4.3 Grafana Dashboard Configuration	5
4.4 Terraform Configuration	5
5. User Interface Overview	6
5.1 Configuring the Variables	6
5.2 View Project Breakdown into Services	8
5.3 View month-wise Breakdown into Services	9
6. Suggestions for Future Work	11
Alternative Design	11
7. Contacting Support	13
Appendix	14
A. Execute Lambda Functions	14

## 1. Introduction

Welcome to the user manual for XC3 Project Breakdown. This document is intended to provide you with all the information you need to get started with and use the feature effectively.

## 2. User Requirements

The user story this document describes, is to view the cost of a project on a more granular level:

*As a user, I want to break down the cost of a project,  
so I can see the cost per service, or the cost per resource.*

## 3. System Requirements and Installation Instructions

There are no additional system requirements or installation steps to use the **‘Project Breakdown Feature’**. It will be deployed as a part of the XC3 solution.

Note: the **‘Project Breakdown Feature’** uses the AWS Cost Explorer API for which you will be charged.

## 4. System Components

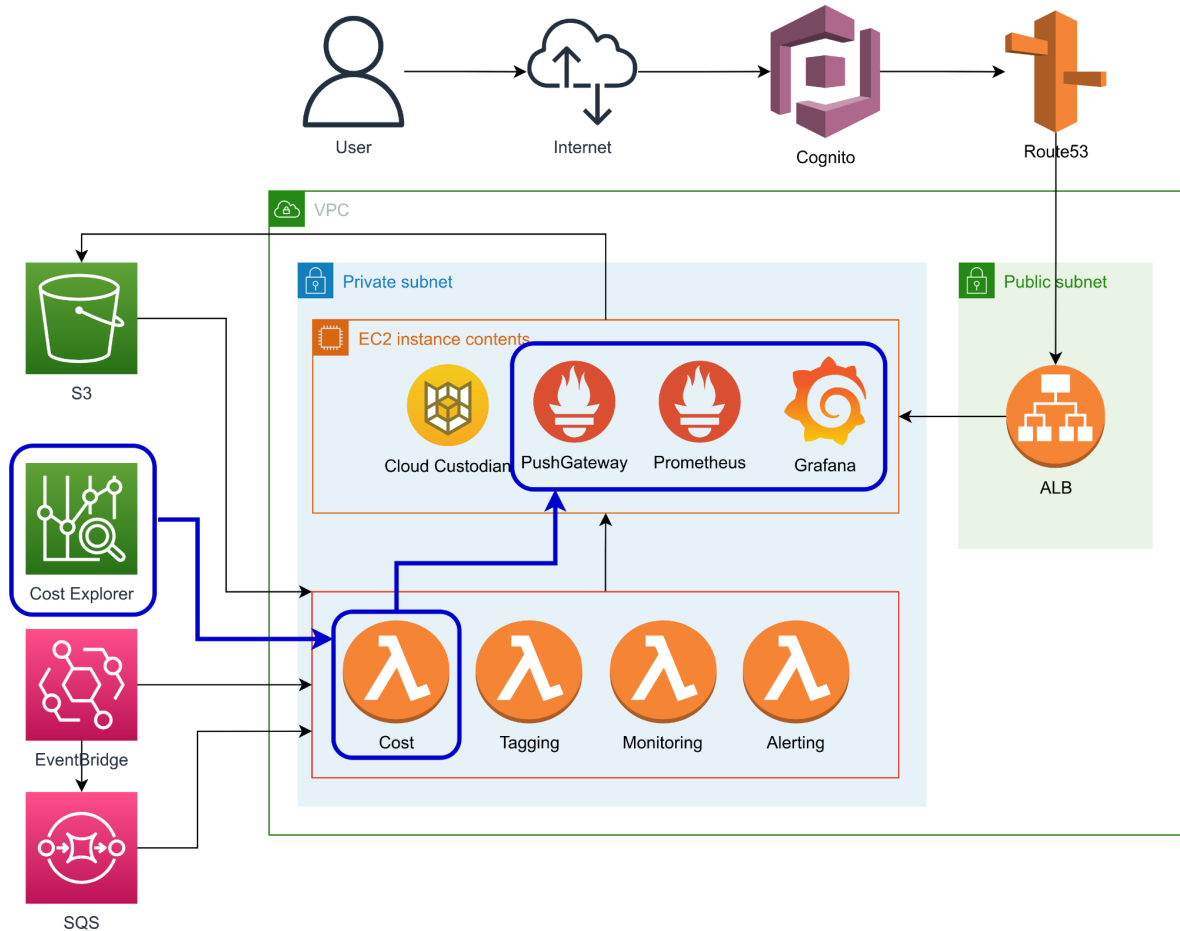
### 4.1 Architecture Overview

The image below shows the architecture of XC3<sup>1</sup>. The project cost breakdown feature only deals with the elements highlighted in blue:

1. Obtain cost data from AWS Cost Explorer API.
2. Process data in the lambda function and push to Prometheus.
3. Show data in a Grafana dashboard.

---

<sup>1</sup> With a few details omitted.



The next sections highlight the function of each part and where they can be found.

## 4.2 Lambda Function

The lambda function '*project\_spend\_breakdown*' is responsible for retrieving and processing the cost data. It uses *get\_cost\_and\_usage\_with\_resources* from AWS Cost Explorer.

**Note:** This service can only obtain the cost data for the last 14 days.

The lambda function is invoked from the *project\_spend\_cost* lambda function:

1. The '*project\_spend\_cost*' lambda function retrieves a list of project tags, a start date and end date.
2. It invokes the '*project\_spend\_breakdown*' lambda function for each of the projects.

3. The ***'project\_spend\_breakdown'*** lambda function uses the ***'get\_cost\_and\_usage\_with\_resources'*** from AWS Cost Explorer to obtain cost information per service and resource.
4. From the response it extracts
  - Month,
  - Service,
  - Resource id,
  - Cost,
5. and pushes the data to Prometheus.

**New:**            **src/budget\_details/project\_spend\_breakdown.py**

**Changed:**    **src/budget\_details/project\_spend\_cost.py**

The cost related to this architecture is \$0.01 per request to AWS Cost Explorer. The response is limited to 8kb. For large projects it might be necessary to use more than one request. The cost estimate is roughly \$0.01 times the number of projects times the frequency of the cronjob for project cost spend.

### **4.3 Grafana Dashboard Configuration**

The Grafana dashboard configuration is added to the *custom\_dashboards* folder. The dashboards will be discussed in section 5.

### **4.4 Terraform Configuration**

Since this feature builds on top of the Project Spend feature, the new lambda function is included in its terraform file.

The file ***'project-spend-lambda-resource.tf'*** has been build upon in the same style as before:

- Added path to lambda python code,
- Add to the lambda function's policy:
  - GetCostAndUsageWithResources,
  - LambdaInvokePermission,
- Add lambda function as a *resource*.

**Changed:** infrastructure/modules/serverless/project-spend-lambda-resource.tf

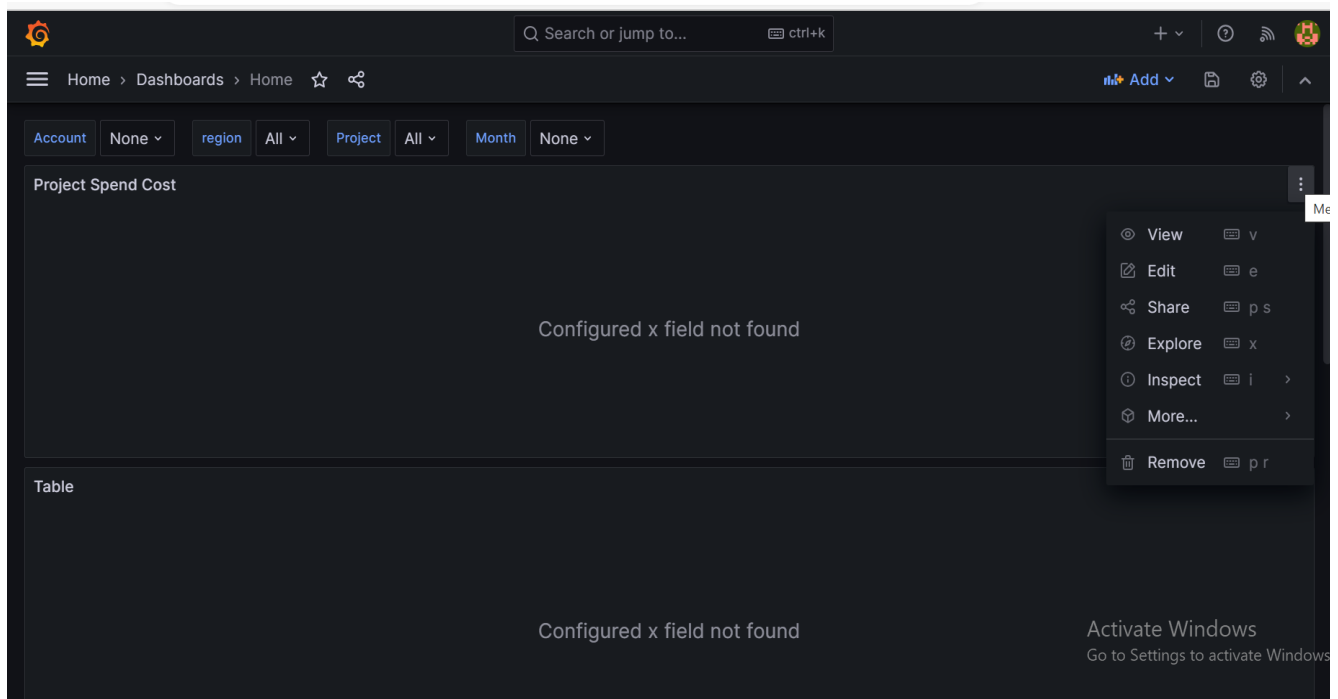
## 5. User Interface Overview

Grafana comes with the new breakdown dashboard pre-loaded.

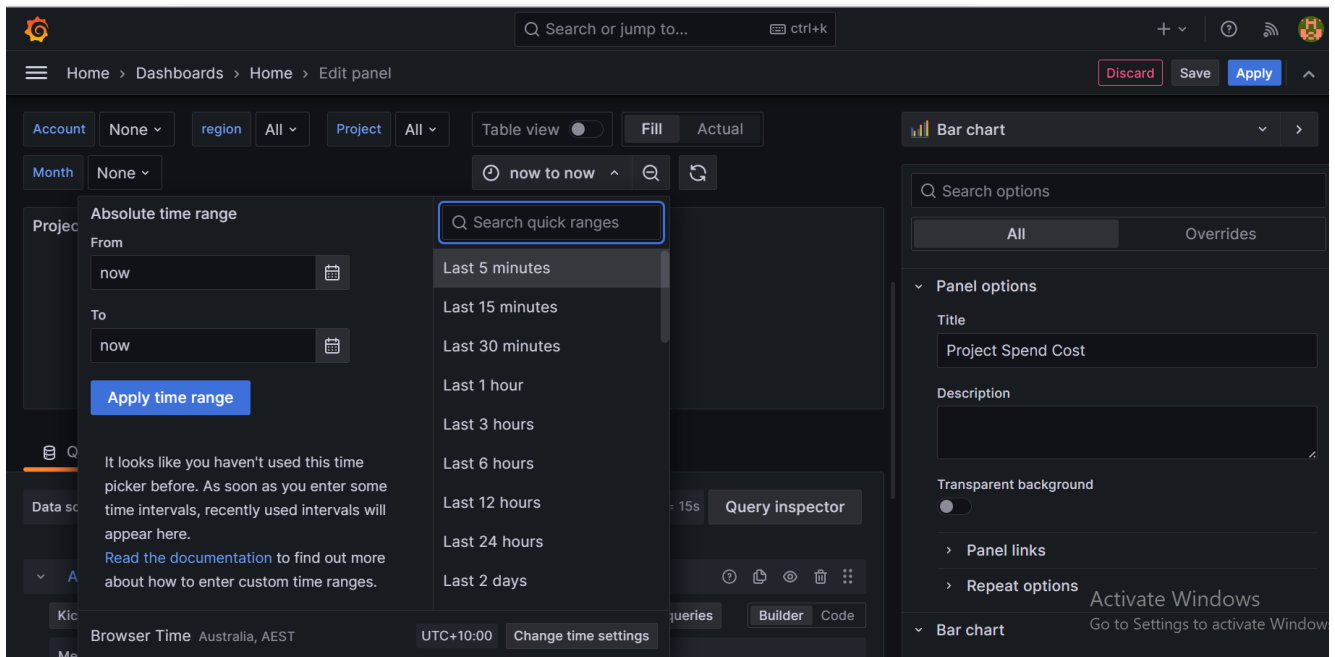
### 5.1 Configuring the Variables

#### Pre-requisites:

- The project spend cost panel should be open.
- Select the three dots from the top right corner of the panel.

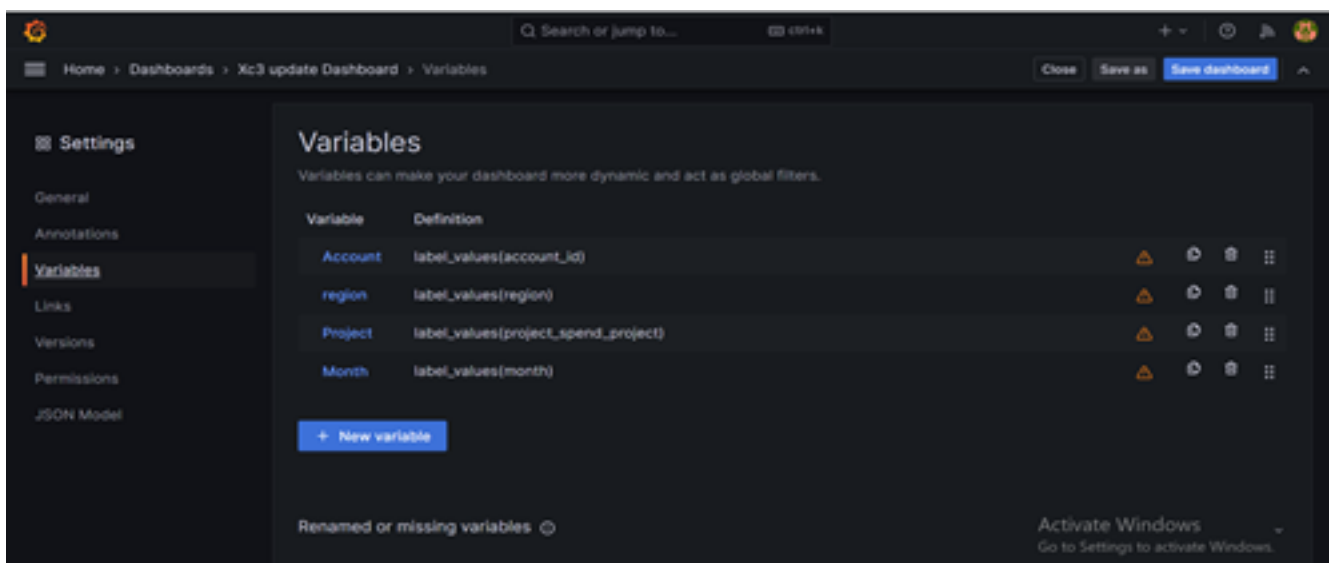


- Set the time from “now to now” TO “within 5 minutes”.



## Steps:

1. Click on the dashboard setting sign on the top right corner of the dashboard.
2. Click on the “variable” option from the left side panel.
3. Click on each variable, navigate to the bottom, and click on the “run query” and the “apply” buttons.
4. Repeat step 3 for each of the variables shown on the dashboard and click on “close”.



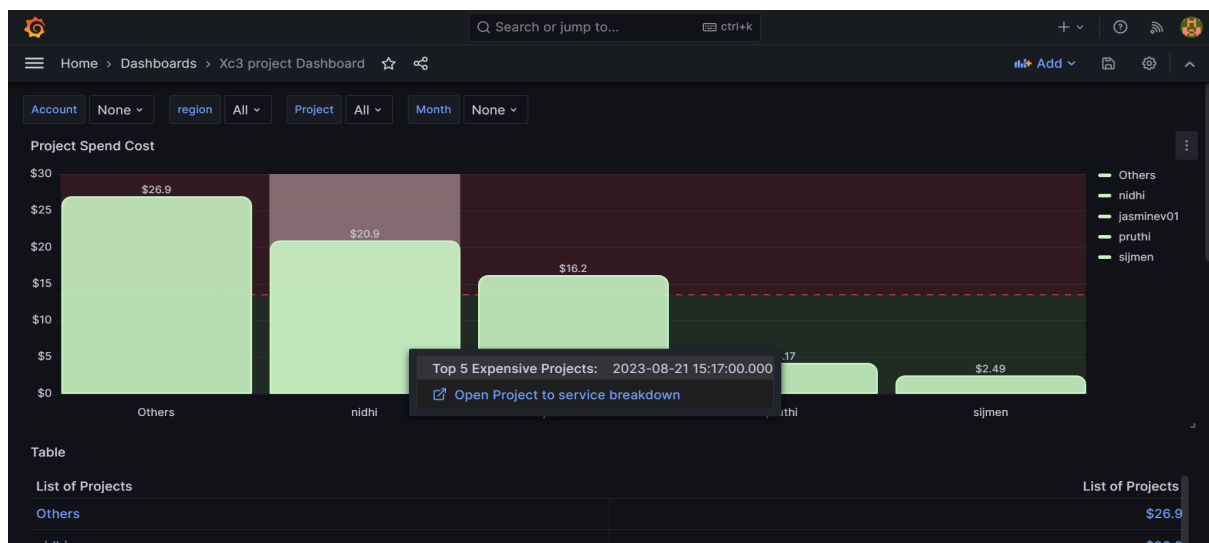
5. After performing the above steps, you can see the metrics on top of the project spend cost panel.

**Note:** The above steps need to be performed if you are unable to see the metrics on the dashboard that you have imported.

## 5.2 View Project Breakdown into Services

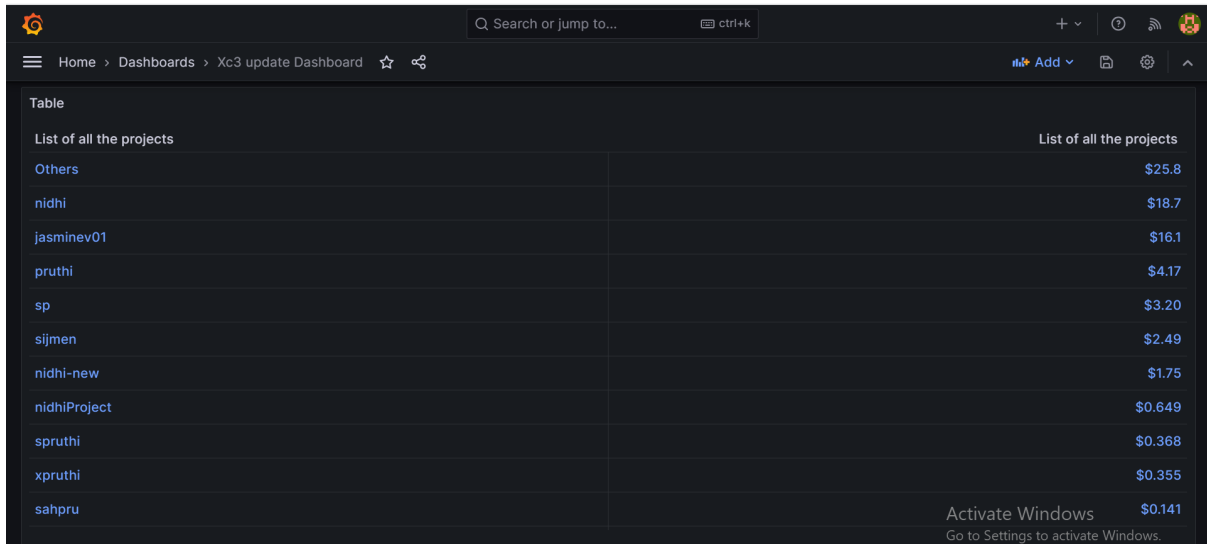
### Steps:

1. On the project spend cost panel, hover on the top of the project bar graph.
2. Click on the link “test open break” that pops up on the screen on top of the bar graph.



The second panel displays the list of all the projects:





Table

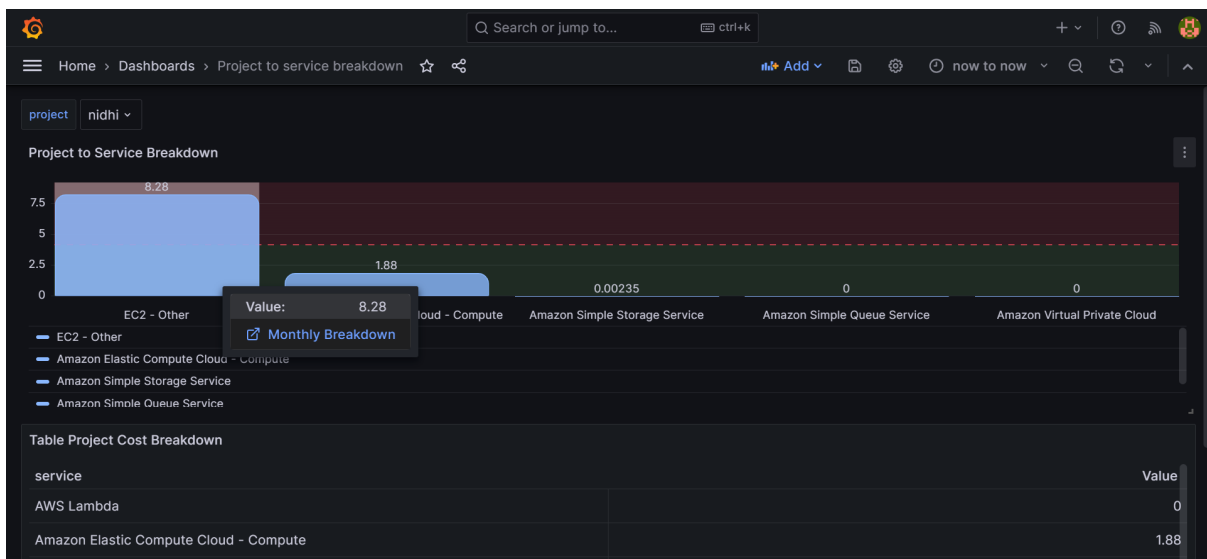
List of all the projects

Project	Cost
Others	\$25.8
nidhi	\$18.7
jasminev01	\$16.1
pruthi	\$4.17
sp	\$3.20
sijmen	\$2.49
nidhi-new	\$1.75
nidhiProject	\$0.649
spruthi	\$0.368
xpruthi	\$0.355
sahpru	\$0.141

Activate Windows  
Go to Settings to activate Windows.

3. It will open a second dashboard “Project to service breakdown” .

- The first panel displays the list of top 5 expensive services.
- The second panel displays the list of all the services.

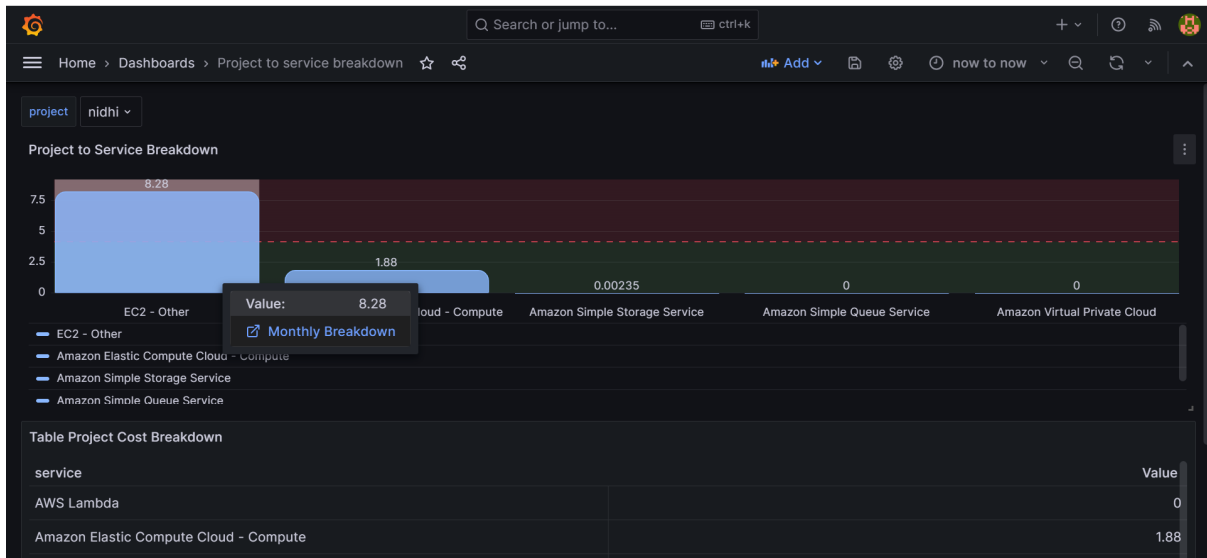


## 5.3 View month-wise Breakdown into Services

### Steps:

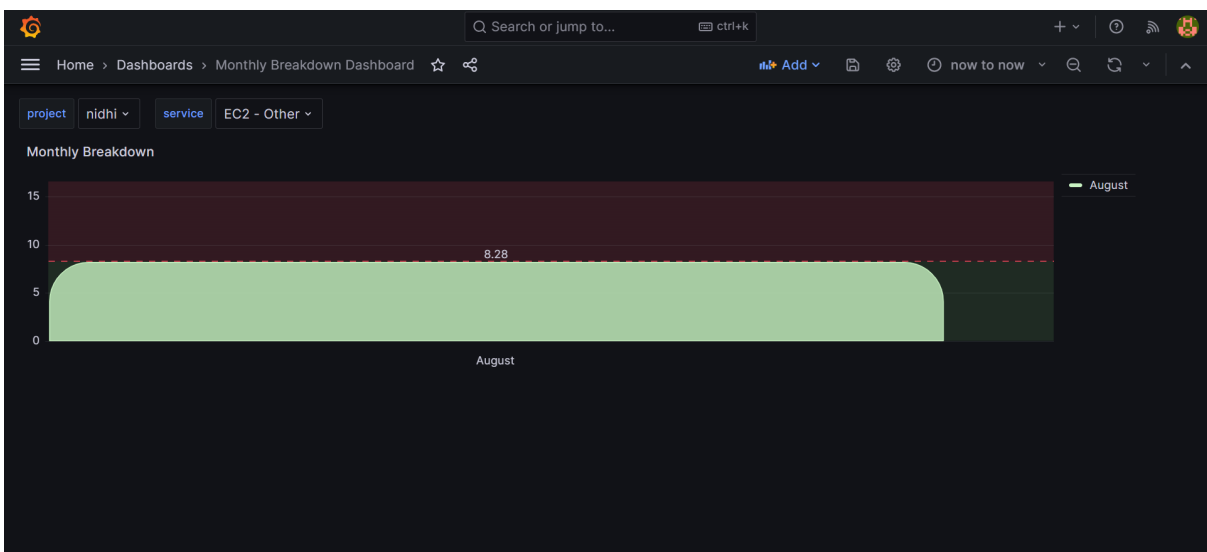
- On the project to service breakdown panel, hover on the top of the project bar graph.
- Click on the link “Monthly Breakdown” that pops up on the screen on top of

the bar graph.



service	Value
AWS Lambda	0
Amazon Elastic Compute Cloud - Compute	1.88
Amazon Simple Notification Service	0
Amazon Simple Queue Service	0
Amazon Simple Storage Service	0.00235
Amazon Virtual Private Cloud	0
EC2 - Other	8.28

3. It will open a third dashboard “Project to service breakdown” .



## 6. Suggestions for Future Work

1. Subsequent calls to terraform plan and apply reveals that terraform changes a lot of resources even when there are no apparent changes to the source files. It seems like it has something to do with the to-be-uploaded zip files. Some sources<sup>2</sup> suggest that this can be fixed by including a hash code in the terraform data blocks. At least during development it might be beneficial if subsequent ‘terraform apply’ invocations have minimal infrastructure changes, so a developer can iterate more quickly.
2. Inside the ‘config.sh’ and ‘init.sh’, if your old variable value is a substring of the variable name, the substitution changes the variable name. For example, an old namespace name of ‘sp’, causes the terraform.auto.tfvars to read `namenewnameace = “newname”`
3. Alerting Mechanism can be done through Grafana Dashboard by doing changes in the Grafana.ini file or making SMTP configuration changes.

### Alternative Design

During our research, we came across an alternative design<sup>3</sup>. This design uses the Cost and Usage Report instead of the Cost Explorer API, see Figure below.

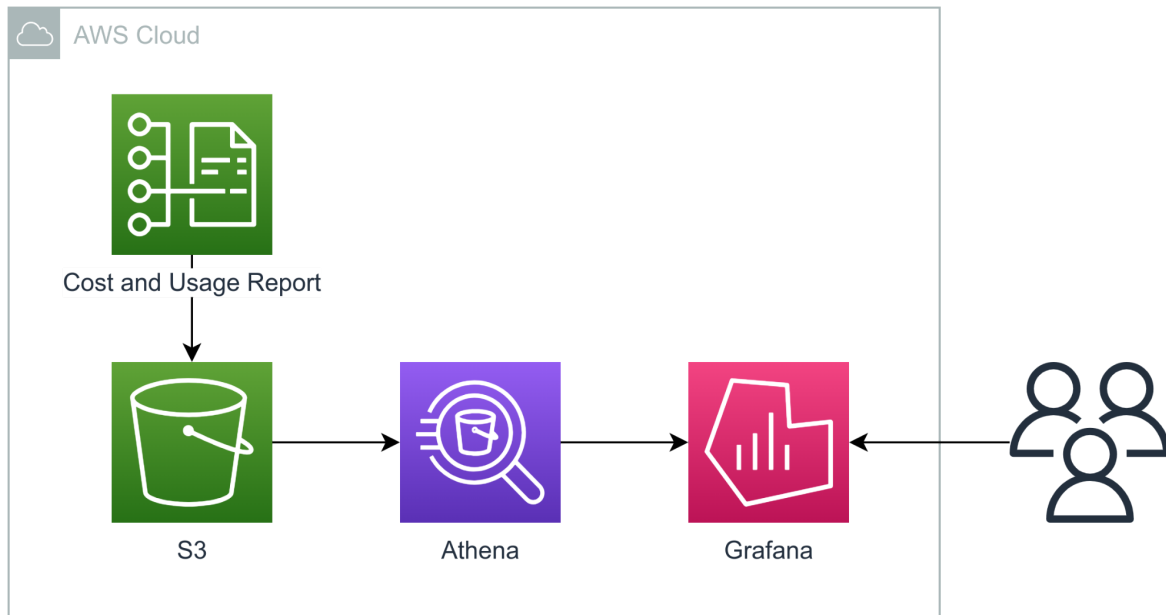
---

<sup>2</sup>

[www.linkedin.com/pulse/lambda-terraform-deploy-only-when-project-changed-cussa-de-souza](https://www.linkedin.com/pulse/lambda-terraform-deploy-only-when-project-changed-cussa-de-souza)

<sup>3</sup>

<https://aws.amazon.com/blogs/mt/visualize-and-gain-insights-into-your-aws-cost-and-usage-with-amazon-managed-grafana/>



The solution works as follows:

1. Set up an S3 bucket as a destination for the Cost and Usage Report (CUR).
2. Set up a Cost and Usage Report<sup>4</sup>.
  - a. The time granularity can be set up as Monthly, Daily or Hourly.
  - b. There is an option to include resource IDs.
  - c. Enable report data integration for Athena. It uses the parquet format.
  - d. The option 'Overwrite existing report' enables you to have the latest copy.
3. Set up an Athena Workgroup that executes the queries from Grafana.
4. Use AWS Glue to create a crawler that creates and updates the Athena table.
5. Configure Grafana to use Athena as data source.
6. Create dashboards in Grafana.
  - a. You can use SQL queries from inside Grafana that will query the data in the S3 bucket through Athena. For example, to obtain the cost per region for the year 2023:

```
SELECT
    product_region,
    sum(cast(line_item_blended_cost as DECIMAL(16,2))) AS cost,
```

---

<sup>4</sup> <https://docs.aws.amazon.com/cur/latest/userguide/creating-cur.html>

```
FROM cur_report
WHERE year = '2023'
GROUP BY product_region;
```

A benefit of this solution is that all data is available in the S3 bucket. Users can change the queries directly from Grafana in case they want to see different reports.

Another positive is that the S3 storage is not limited to 14 days of cost data, unlike the Cost Explorer API. As long as the S3 bucket is not deleted, the cost data will be accumulated.

A possible drawback of this solution is that it might take up more storage. It could be AWS specific. If XC3 likes to report cost data from other cloud providers, there might need to be an intermediate step to process the cost data in a common format.

It is unclear which solution is more cost effective. CUR reports incurs cost per resource per time unit, while AWS Cost Explorer API incurs a cost per request.

## 7. Contacting Support

Lastly, the user manual contains detailed instructions and visual aids to help users install, configure, and use the Xgrid's XC3: Project Cost Breakdown Feature. In case of any queries get in touch with Team 14 at Swinburne University for further assistance.

## Appendix

### A. Execute Lambda Functions

As a developer who is testing the infrastructure, you need to execute the lambda functions manually after deployment. The following bash script can be used to execute the necessary lambda functions:

---

```
prefix=my-namespace
function_names=(
    "-list_linked_accounts"
    "-total_account_cost"
    "-project-spend-cost"
    "-most_expensive_service_lambda"
)
for name in "${function_names[@]}"
do
    aws lambda invoke
        --function-name $prefix$name out
        --log-type Tail
        --query 'LogResult'
        --output text
    | base64 -d
done
```

---