

****

author(s) : Narsing Mishra

version : 1.0

status : Reviewed

source : EVIDEN

date : 23/10/2023

pages : 47

owner : Narsing Mishra/Siara Jeyaraj

List of changes

| Version | Date | Description | Author(s) |
| --- | --- | --- | --- |
| 0.1 | 29-06-2023 | First draft issue of the document | Narsing Mishra |
| 1.0 | 09-07-2023 | Final draft issue of the document | Siara Jeyaraj |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Table of Contents**

[1. About this Document 4](#_Toc140592767)

[1.1 Document Information 4](#_Toc140592768)

[1.2 Distribution List 4](#_Toc140592769)

[1.3 Review Summary 4](#_Toc140592770)

[2.0 Eviden Carbon Calculator 5](#_Toc140592771)

[2.1 Benefits 5](#_Toc140592772)

[2.2 Use Cases 5](#_Toc140592773)

[2.3 Features 6](#_Toc140592774)

[3.0 Inventory Management 6](#_Toc140592775)

[3.1 Taxonomy Management 7](#_Toc140592776)

[3.4 User Management 7](#_Toc140592777)

[3.5 Dashboard Features 7](#_Toc140592778)

[4.0 Architecture 7](#_Toc140592779)

[4.1 Solution Architecture 7](#_Toc140592780)

[4.2 Architecture of the current scope phase. 8](#_Toc140592781)

[4.3 Application Data flow 8](#_Toc140592782)

[4.4 Agreed deliverables 9](#_Toc140592783)

[5.0 QuickSight Dashboard 10](#_Toc140592784)

[5.1 User Experience for ECCA Application 10](#_Toc140592785)

[5.3 Pie charts 18](#_Toc140592786)

[6.0 AWS services & 3rd Party Software used 19](#_Toc140592787)

[6.1 Open Source or 3rd Party SW list 20](#_Toc140592788)

[6.2 Deployment Model 20](#_Toc140592789)

[6.3 Deployment Options 20](#_Toc140592790)

[6.4 Lead Time For Deployment 20](#_Toc140592791)

[6.5 Supported Regions 20](#_Toc140592792)

[6.6 Technical Pre-requisites 20](#_Toc140592793)

[6.7 Technical Skills Required 21](#_Toc140592794)

[6.8 Environment Configuration 21](#_Toc140592795)

[7.0 Security 21](#_Toc140592796)

[7.1 Deployment Privileges 21](#_Toc140592797)

[7.2 IAM Roles and Policies 21](#_Toc140592798)

[7.3 Authentication & Authorization 22](#_Toc140592799)

[7.3 Encryption 22](#_Toc140592800)

[7.4 Secrets Management 22](#_Toc140592801)

[7.5 Customer Data 22](#_Toc140592802)

[7.6 Network Configuration 22](#_Toc140592803)

[8.0 Cost 23](#_Toc140592804)

[8.1Service Costing 23](#_Toc140592805)

[8.2 Solution Costing 24](#_Toc140592806)

[8.3 Solution Sizing 24](#_Toc140592807)

[9.0 Installation / Deployment 24](#_Toc140592808)

[9.1 Step by step deployment 24](#_Toc140592809)

[9.2 Post app installation (Validation of DNS with AWS SSL certificate) 39](#_Toc140592810)

[10.0 Troubleshooting 40](#_Toc140592811)

[10.1 Scaling 43](#_Toc140592812)

[10.2 Support 43](#_Toc140592813)

[10.3 Scope 43](#_Toc140592814)

[10.4 Support duration 43](#_Toc140592815)

[10.5 Support Plan 43](#_Toc140592816)

[10.6 Contact 44](#_Toc140592817)

[10.7 Backup & Recovery 44](#_Toc140592818)

[11.0 Routine Maintenance 44](#_Toc140592819)

[11.1 Key & Credentials Change 44](#_Toc140592820)

[11.1 Software Updates 44](#_Toc140592821)

[11.2 License 44](#_Toc140592822)

[11.3 Pricing Models: 44](#_Toc140592823)

[11.4 Service Limits 45](#_Toc140592824)

# About this Document

## 

## 1.1 Document Information

| Project Name: | eviden | | |
| --- | --- | --- | --- |
| Document Reference Number: |  | Document Version No: | 1.1 |
| \*Request/RFC/CCN Number (If any): | N/A | Document Version Date: | 22-03-2023 |
| Author | Narsing Mishra | | |

## 

## 1.2 Distribution List

| Name | Role | Action | Owner |
| --- | --- | --- | --- |
|  | Product Manager | Review and approve |  |
| Manish Gupta | Product Architect | Review and approve |  |

## 

## 1.3 Review Summary

There are several areas that require close attention during the review of this design. These include but are not limited to:

|  |  |  |  |
| --- | --- | --- | --- |
| Document Section | Technology | Tower | Description |
|  |  |  |  |
|  |  |  |  |

# 2.0 Eviden Carbon Calculator

Carbon Calculator is a platform that will enable organizations to measure, report their carbon footprint. The platform is built on AWS. The multisided platform will be made available across different geographies, depending upon the need of the businesses and governments, specific to data security and potential latency and availability-based requirements.

Any specific regional policy related to local jurisdictions and constraints for IT systems deployment or data localization requirements will be handled on the case-by-case basis to meet the global and/or local requirements.

The users of the platform will be organization’ sustainability teams i.e., Corporate Social Responsibility and executive (CxO’s) teams, persons associated with carbon regulatory reporting and disclosures, 3rd party auditors and accredited third parties.

By design, the platform will extend and scale to create multi-sided solution, several features are under construction and will be further extended for automated advisory for businesses and government, using advanced Artificial Intelligence and Machine Learning (AI/ML) technologies.

Vision under construction is to have a multi-sided platform to support climate advisory, measurement and management of carbon footprint and to build projections to ensure businesses reach their carbon pledges and targets.

This to ensure that we work collectively to save the planet and in turn reduce the risk of climate change for sustainable ecosystem, supporting biodiversity and life on the planet, including human beings.

## 2.1 Benefits

* It accepts IT assets inventory data simple input template.
* It calculates scope 2 and scope 3 carbon emissions for IT assets inventory and provide GHG reporting.
* The output on dashboard can be filtered on various criteria like Category, region, batch id etc.
* It allows three different users levels for accessing the application. Superuser, Data Manager and General user.

## 2.2 Use Cases

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Use Case(s)​** | | **Role(s)​** | **Customer Issues​** | **How Does the solution address the Issue?** | |
| Establish a baseline of their IT emission, and producing carbon reduction plan based on hotspots identified within the emission landscape​ | | CIO  IT Manager  CTO  Sustainability Officer​ | | Adaptable IT assets carbon footprint calculation platform which can revise the carbon (without huge manual efforts) footprint based on the changes in the underlying asset.​ | The Carbon Calculator for IT estate, with the help of Climate Advisory team allowed to align the reporting calculations to GHG protocol​  Data was collated and transformed from four different regions (globally), and persisted in database​  Reporting - Carbon footprint of IT assets for cradle to gate (manufacturing) and carbon footprint usage during the lifetime​ |
| Calculate Emission of end users' devices and provide visibility of emission source​ | | CIO  IT Manager  CTO  Sustainability Officer​ | | Need to view of the emission associated with EUCs and understand the emission sources.​ | The Carbon Calculator was used for End User Compute​  Reporting the emissions of end user devices​  The dashboard allows the visualization of emission hotspots​ |
| Provide a carbon impact simulation of each digital transformation project for prioritization.​ | | IT Manager  IT Transformation Director  CTO​ | | Evaluating the results of IT transformation projects and need visibility of the carbon impact of projects to help prioritize the transformation activities.​ | Migration of SAP workloads into the Public Cloud, baseline, and revised baseline of the carbon footprint.​​  Help customers to understand the carbon impact of each transformation project against the baseline. ​​  Visualization of carbon footprint of individual project to prioritize the transformation activity.​​ |

## Features

Below are some worth mentioning features of ECCA.

There are 3 user roles defined in the application as a hierarchy of rights, all requiring a login and a password:

* The Super Admin is the role to manage the application and all its features
* The Data Admin can upload inventories and see the results in dashboard.
* General users can see the results of computations in the dashboard.

The Carbon Calculator offers several features which we can classify in the following 4 categories detailed below.

But a high-level picture of the application is the one of a calculator matching inventory items with the corresponding emission factors in a taxonomy to yield carbon footprint impact for a given period.

## 3.0 Inventory Management

The Inventory

The inventory lists all the items considered in the computation. Initially our calculator is designed to calculate the ICT carbon footprint but will be extended to other market specific items.

Data admins are expected to upload inventory files in the system, once convinced of the upload quality, insert these in the database to allow the computation. A batch process is started to make all the calculations and once done, the result can be visualized in the dashboard.

Here is an explanation of the different fields required to define an inventory.

Inventory\_ID: a key allowing the grouping of items under a common value. This key will allow the selective display of the inventory in the dashboard. Usually represents the date of the inventory upload.

Item Type: determines the type of item. Examples are laptops, servers, …

Supplier: The supplier of the inventory item.

Model: the model of the item of the supplier.

Quantity: the number of the items with all the other fields equal.

Country: the country where the item is located

Equipment\_Procurement\_Date: the date the item was first used in the organisation as a new device (so not its later second hand uses). At this date, the computation will take into the embodied part of the item production and transport.

Depreciation\_Start\_Date: the expected date when the item is planned to be decommissioned. At that time, the item will not be considered in the computation.

Ownership: is the item owned or leased

User has the possibility to add up to 3 additional columns which will be transferred to the dashboard but not considered in the computation. It allows the user to filter or dig by additional, custom dimensions.

As inventory uploads are made in batch, there is a batch report available to monitor the progress.

## 3.1 Taxonomy Management

The Taxonomy contains the emission factors of items found in the inventory. According to the level of detail of the taxonomy (which is enriched regularly by Atos and by the users via specific upload by a data admin) the computation will use in first place (unless otherwise stated) the customer specific data, then the detailed individual factors matching items and finally for the missing ones the default market standard emission factors for the given item-type.

According to the license model, it is possible for a data admin to upload company/industry specific emission factors for items and items-types not known to Atos. We can give the example of a Japanese company which has detailed Sony Xperia emission factors and wants to upload these in the taxonomy.

## 3.4 User Management

The application is initially provided with 1 Super Admin who has all the rights (= is allowed to activate all the features). Once he has connected and updated his password, the Super Admin can create additional users. The process is the following

* Add user record
* Fill in the user details, including email, username.
* Save the changes.
* The tool will send an email to that user with an application link with their initial password.
* Now user can access according to the role he was granted.

The Super Admin can then add other users, edit or deleting existing ones.

## 3.5 Dashboard Features

The Dashboard has several components:

* A summary band consolidating the most important KPIS of the computation
* A selection band to allow filtering and digging.
* A Pie chart to view the relative importance of the items in some selected dimensions
* A geographic spread of the computed impacts
* A table computing totals by emission factors and risks
* A graphical view of the impacts.

Once at least one inventory is uploaded, the dashboard will show the result of the computations of all the uploaded dashboards. It is also possible for the user to select 1 or several inventories to be displayed and once displayed, the possibility exists to filter per value of the selected dimension.

# Architecture

## Solution Architecture

The ECCA application is split in two parts:

* A back-end server running Java Spring Boot framework
* A front-end running Angular.
* A database RDS PostgreSQL

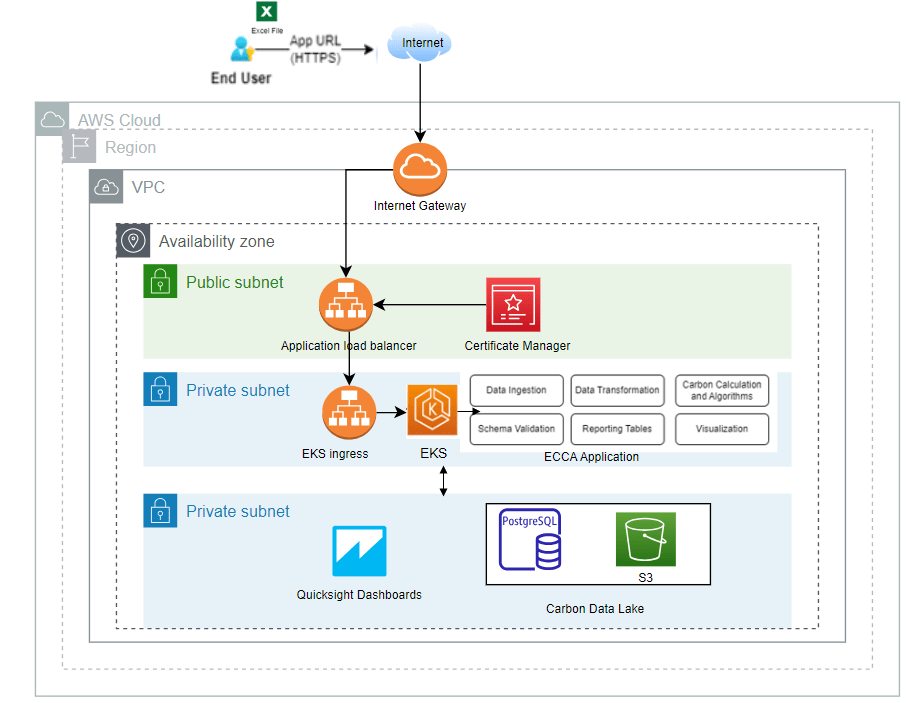


Figure 1 ECCA Architecture

## 4.2 Architecture of the current scope phase.

The Carbon calculator application which is containerized will be deployed on EKS cluster using the Terraform scripts hosted on git repository. The internal tasks reside on private subnets and won’t be exposed to public internet. The application is DNS registered and uses the ssl certificate

Carbon Calculator is an application built based on Angular Framework and API is built using Java Spring boot framework. Frontend is built using Angular Framework which uses typescript, html, and CSS. Both the front-end and the back-end components of Carbon Calculator are containerized image respectively. The front-end components are exposed using Application Load balancer. The user management required for Carbon calculator application are provisioned within Database itself.

## 4.3 Application Data flow

The end user logs in with username and password; once authenticated JWT token gets created by backend java spring boot API and then token get assigns to the user by which the application can be accessed. The user can then upload the MS Excel file containing IT assets. The data validation checks on the input data are done by Atos Carbon Calculator application, and in case of valid input, data is stored in S3 bucket and then S3 bucket data is copied to RDS PostgreSQL via Lambda function. The data will further be processed, and carbon footprint will be calculated based on business logic and the resulting carbon calculator footprint data for reporting is stored back in RDS PostgreSQL.

Next Carbon calculator feature is to visualize the carbon footprint in QuickSight dashboard to the end user. Brief about the same is explained further in this document. This feature picks the data from RDS PostgreSQL and presents the end user with the resulting Dashboard for its inventory data.

User management feature available in the user management section of the ECCA application, where the superuser can add, edit, or delete users specific to the customer domain. User registration, login and password details are managed using Forgot Password feature and default username/password is made available to the user via email.

The certificates management is achieved through AWS certificate manager. The security management is achieved using the JWT tokens (user authentication), Digital certificates, along with AWS native Security features. For securing the application, we use Security groups on Load balancer and in between the EKS, S3 and RDS PostgreSQL.

## 4.4 Agreed deliverables

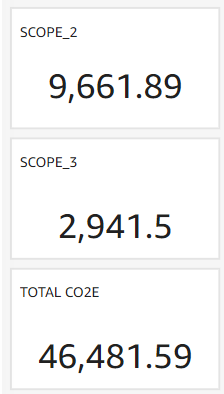
IT Carbon Calculator Phase 1 deliverable (highlighted in blue in the table below).

|  |  |  |  |
| --- | --- | --- | --- |
| **Phase** | **Function Scope/Description** | **Timeline** | **Deliverable(s)** |
| **Phase 1: MVP** | Carbon calculator for IT Assets – Functionality (like for like- as is porting) as current capability | Jan 2023 – Mar 2023 | Demo, Solution brief, First Call Deck, SDR |
| **Phase 2: MVP-Enhancement** | **Carbon calculator for IT Assets – with Reports and Dashboard Enhancements** | **Mar 2023 – May 2023** | **Deployment Guide, Demo, Solution brief, First Call Deck,** |
| **FTR (Foundational Technical Review)** |
| **Phase 3: Integration** | AWS Carbon Calculator Integration + Cloudamize | Jun 2023 – Aug 2023 | Deployment Guide, Demo, Solution brief, First Call Deck, |
| FTR (Foundational Technical Review) |
| **Phase 4: Integration** | Integration with SaaS providers (SAP Concur + ServiceNow CMDB) | Sep 2023 – Nov 2023 | Deployment Guide, Demo, Solution brief, First Call Deck, |
| FTR (Foundational Technical Review) |
| Creation of #25 SFDC Opportunities by end of Phase 4, resulting in a Total pipeline Value of $7.5Mn TCV |
|  |
| **Phase 5: Financial Services Industry** | Financial Services - Mortgage Portfolio | Dec 2023 – Mar 2024 | Deployment Guide, Demo, Solution brief, First Call Deck, |
| SDR |
| **Phase 6: Financial Services Industry** | Financial Services - Investment Portfolio | Mar 2024 – May 2024 | Deployment Guide, Demo, Solution brief, First Call Deck, |
| SDR |
| **Phase 7: Telecom Industry** | Telecom Sector - 5G Infrastructure (through IoT) PoC/ MVP | Jun 2024 – Sep 2024 | Deployment Guide, Demo, Solution brief, First Call Deck, |
| FTR (Foundational Technical Review) |
| Creation of #25 additional SFDC Opportunities by end of Phase 7, resulting in a Total pipeline Value of $7.5Mn TCV |
| Deployment Guide, Demo, Solution brief, First Call Deck, |
| **Phase 8: Energy & Utilities Industry** | Energy & Utilities - Energy production, distribution, and usage (TBD) | Sep 2024 – Nov 2024 | FTR (Foundational Technical Review) |
|  |

# 5.0 QuickSight Dashboard

Amazon QuickSight is a business analytics service provided by AWS. QuickSight provides easy to use tools to build visualizations, perform ad-hoc analysis, get business insights from the data, and share the results with others.

Key carbon KPI’s are provided for business to make informed decisions



## 5.1 User Experience for ECCA Application

Screen shots of ECCA application for authorised user to guide, and help better User experience, use case is Carbon Footprint Calculating step by Step guide screen shot below. Once the application and it is underlying infra is installed in respective customer AWS account, the customer must register the application URL with DNS and get SSL certificate, the application name can be prefixed to their DNS for ex in this example ECCA.myco2c.adh is prefixed to atos.net.

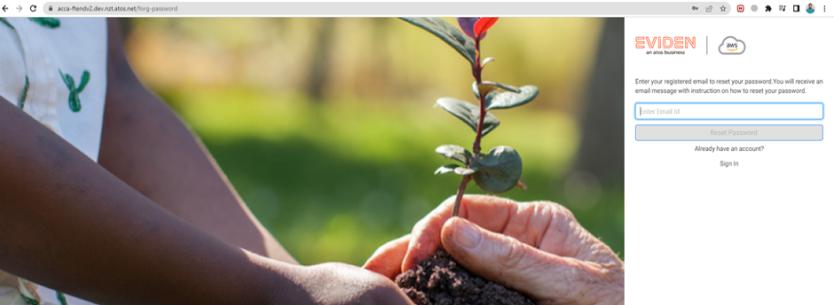
Click on the URL [ECCA.myco2c.adh.atos.net/index](https://acca.myco2c.adh.atos.net/index) and it will land below page:



If user enters wrong credentials, it gives below error:



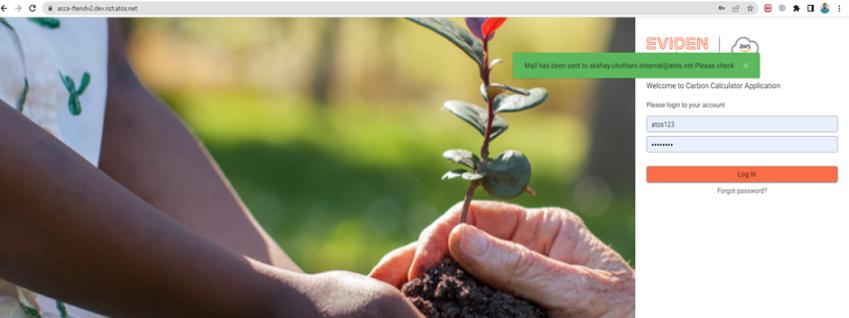
If user forgot the password, user need to click on forgot password link it will redirect to forgot password page.

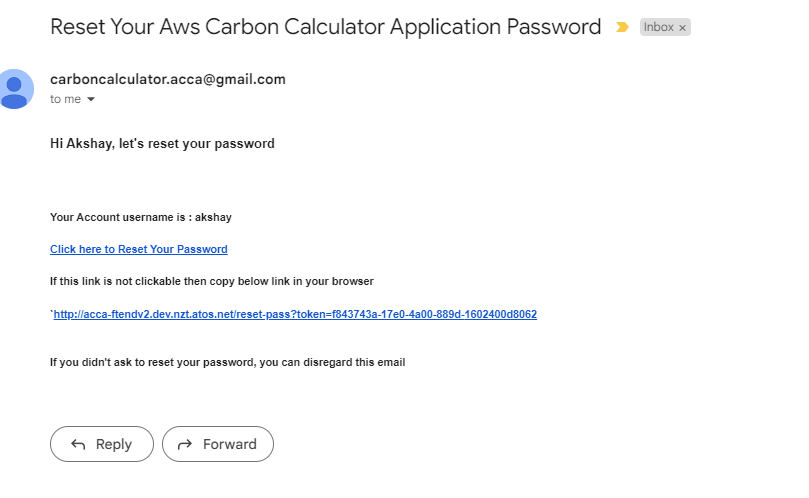


After that user need to enter his registered email address associated with his account and if user enters wrong email, it will give below error.



If users enter correct email address, User will receive the mail on his registered email address.

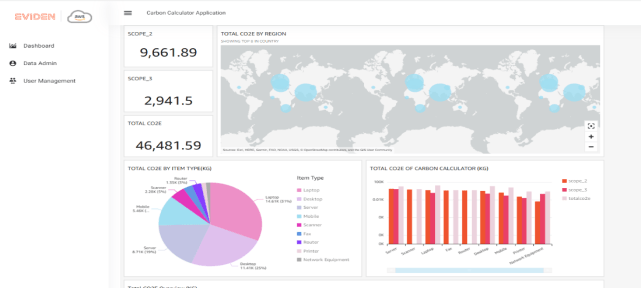




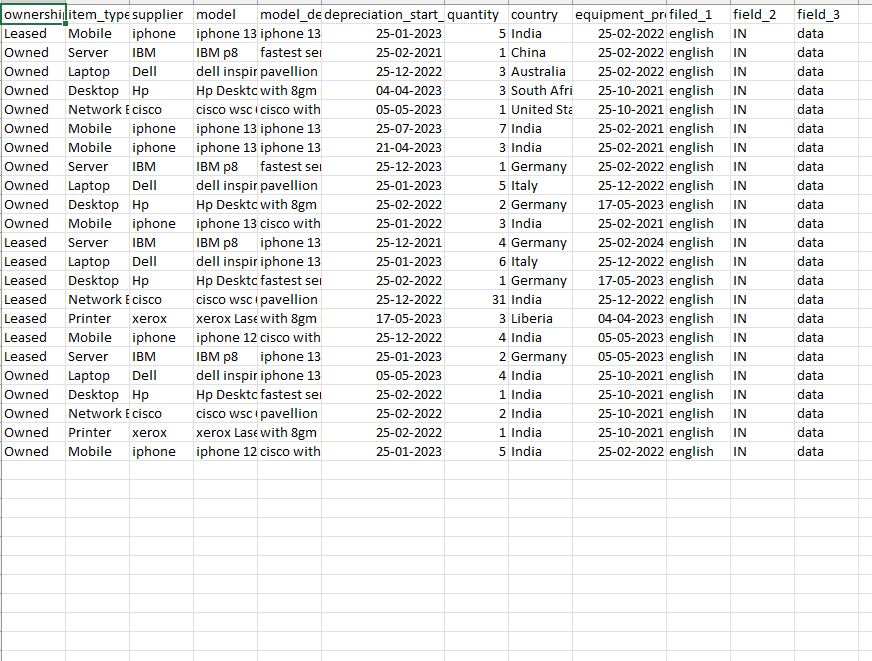
After that user can navigate to Application and enter the credentials.

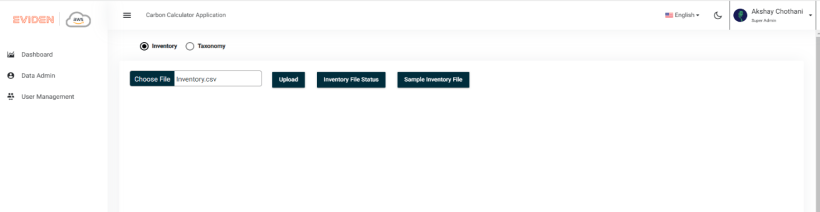


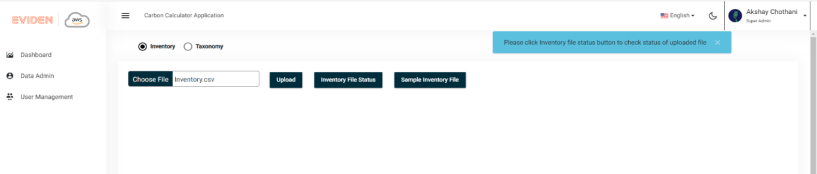
After logging in Dashboard page(home) page will be displayed as shown below:



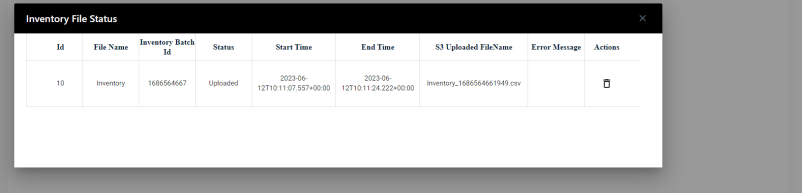
Once user clicks Data Admin tab from menu, he will be redirect to Inventory page and then select appropriate inventory file and click on upload button.



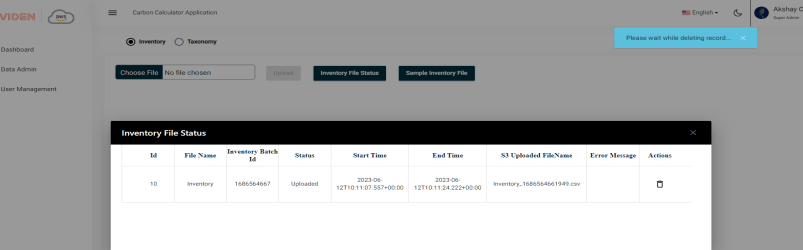


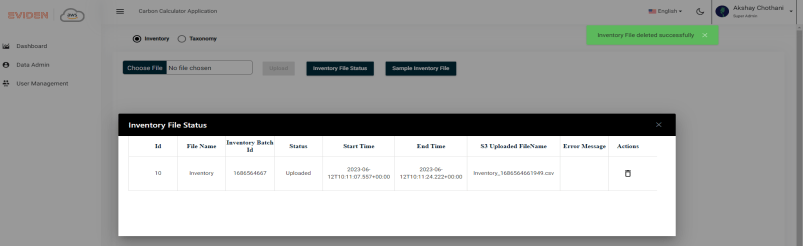


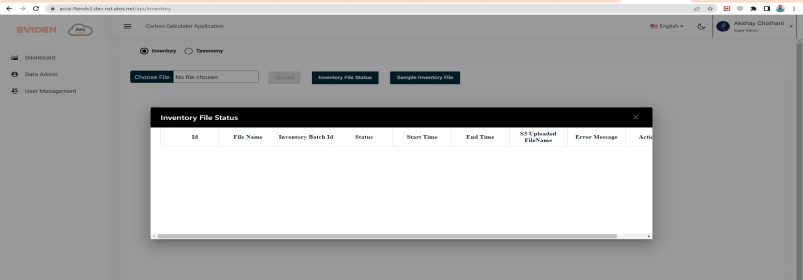
Wait for some time, uploaded file information will be displayed in Inventory File Status popup



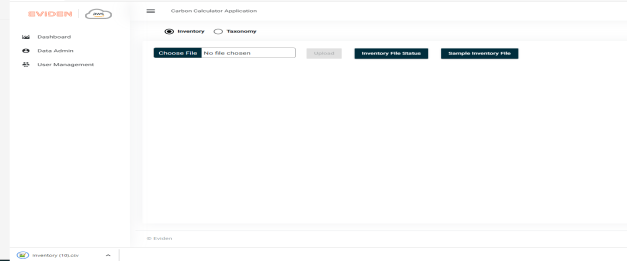
User can delete an unwanted uploaded file via delete button in action tab





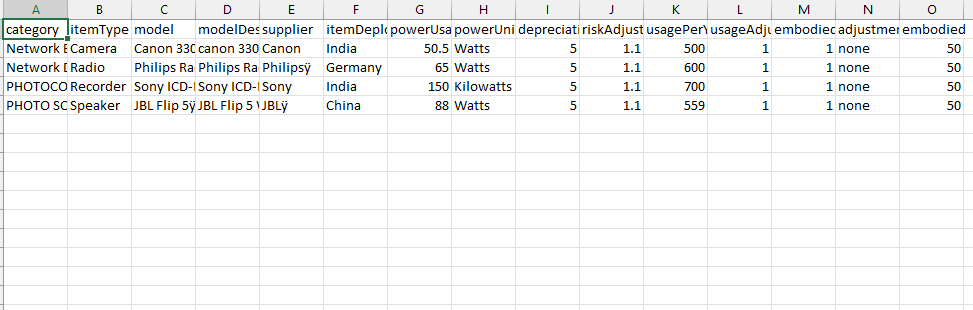


User can download sample Inventory File using Sample Inventory File button

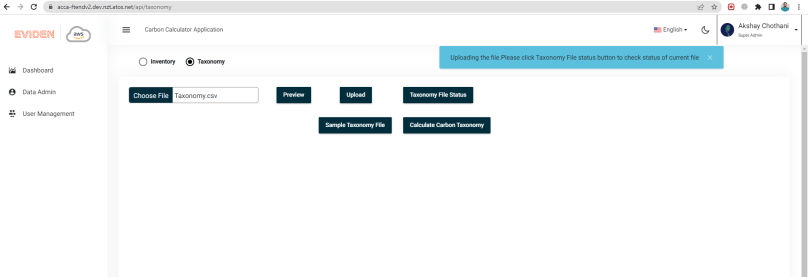


Taxonomy Tab

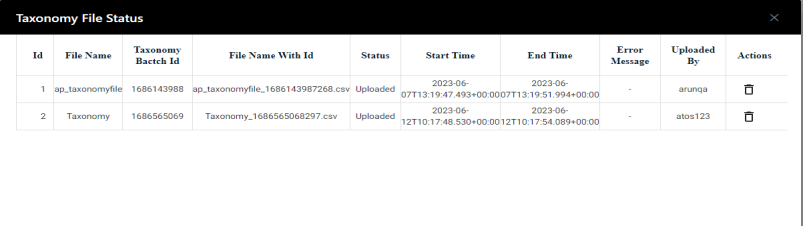
Sample taxonomy file looks like below:



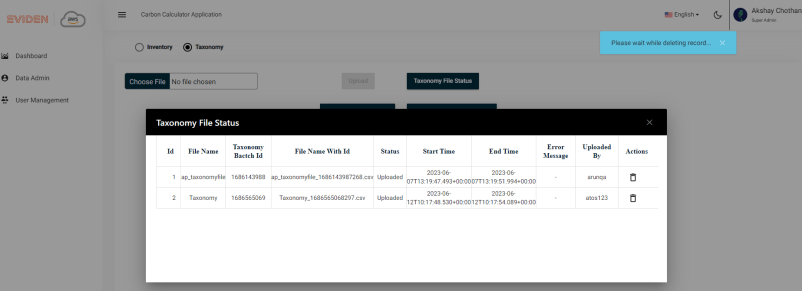
User can upload taxonomy file to upload. Preview buttons look like below:



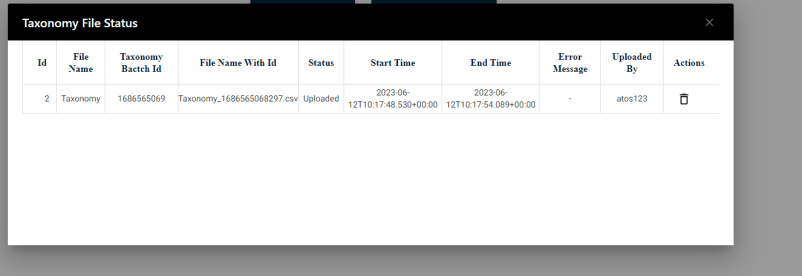
User can view the uploaded file in taxonomy file status popup.



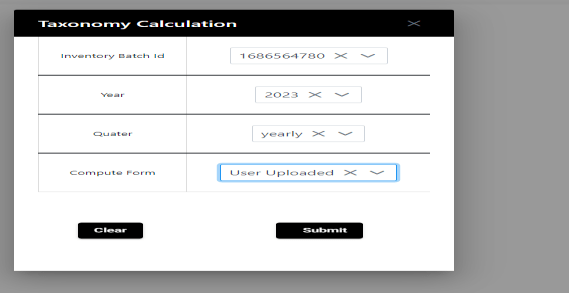
User can delete the unwanted taxonomy file



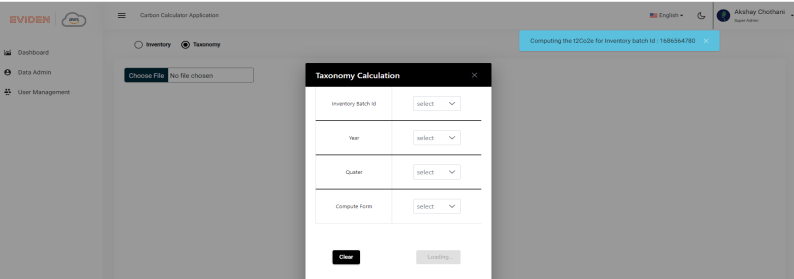
Record 1 is deleted.



In calculate carbon taxonomy user needs to enter Inventory file batch id and other details.

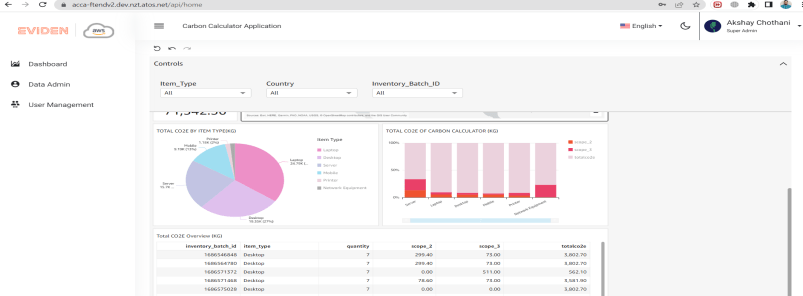


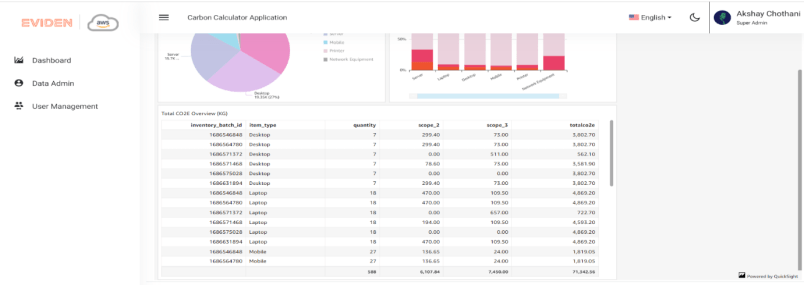
Once user clicks on submit tco2e will be calculated for given inventory batch id.



Once it is computed user can redirect to dashboard screen to view the results(computation) in graph and chart formats

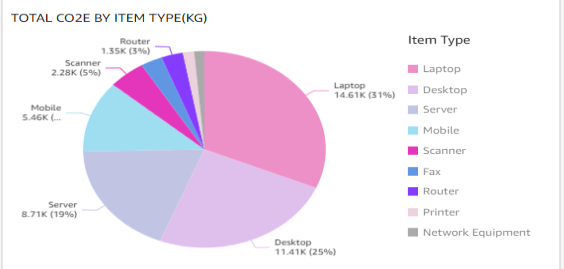
5.2 Carbon Foot Print Dashboard



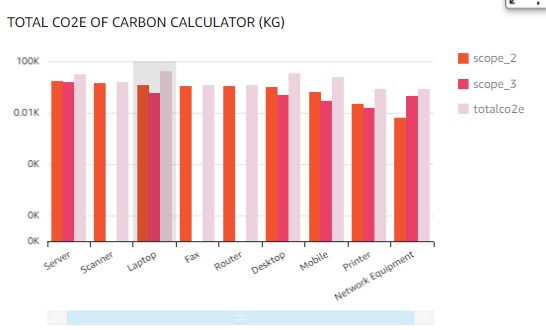


## 5.3 Pie charts

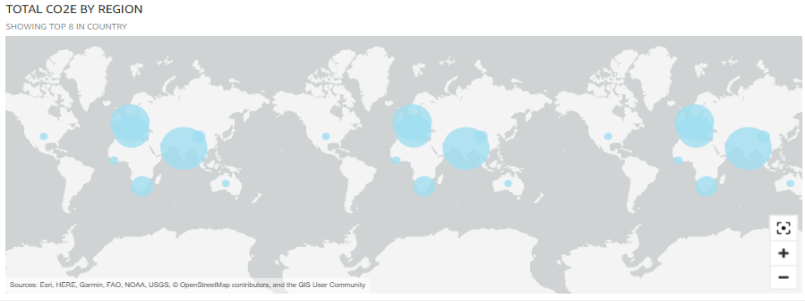
Pie charts represents for Total CO2e usage by item type (KG).



Bar chart represents count of scop2 and scope3 and Total CO2e for Each Item category such as Printer, Mobile, Desktop etc.



Map chart represents the tcO2e region wise.



# 6.0 AWS services & 3rd Party Software used

The ECCA application utilises AWS services as mentioned below in table 3. The third-party software used are mentioned as below in table 4.

|  |  |  |
| --- | --- | --- |
|  | **AWS SERVICES Used** | **Resources created** |
| AWS EKS Cluster | EKS Cluster |
| AWS IAM | 2 IAM Roles with permission listed in section 3.2 |
| AWS VPC | 1 VPC with 3 public and 3 private subnets |
| AWS Security Group | Four Security Groups created   * ECS Security Group, * RDS Security Group, * PostgreSQL Group and * Load Balancer Security Group |
| AWS Load Balancer | 1 Public Load balancer |
| AWS ECR | Repositories to house the application images |
| AWS QuickSight | Quicksight dashboards are bundled and Display it in frontend. |
| AWS RDS | Amazon RDS for PostgreSQL |
| ECS Application | ECS task and service |
| AWS Secrets Manager | one Secrets created  PostgreSQL |
|  | AWS NAT Gateway | 1 NAT Gateway |
|  | AWS ACM | 1 Aws Certificate Manager |
|  | AWS KMS | AWS Managed Key for |
|  | AWS Market Place | For hosting the ECCA application |
|  | AWS License Manager | For License management of ECCA app |
|  | AWS Cloud Watch | For Monitoring Ec2 instance |

Table 3

## Open Source or 3rd Party SW list

#### List of software used by ECCA Core

Following software has been used with core ECCA application to support the functionality of managing AWS Media Services from the user interface.

|  |  |
| --- | --- |
| **Software** | **Version** |
| Angular | 17.0.41 |
| Spring Boot | 6.0 |
| Java | 17 |
| Kubernetes | 4.9.1 |
| NGINX | 1.23.0 |
| PostgreSQL | Latest |
| Tomcat | 10.1.7 |

Table 1

## Deployment Model

The ECCA application in containerized image is available in AWS marketplace container registry, The code for Infra and application to be deployed is available in git repository. The customer should have AWS account as prerequisite to purchase the application. After which the application with underlying Infra can be deployed in the AWS account. This solution is designed to be deploy both infra and application in AZ to achieve good performance. In case if customer wants this in multi AZ then TF code has to be modified.

## Deployment Options

The buyer will get the URL for git repository (contains infrastructure code to generate the environment) and deployment steps while they purchase the application. Once the Infra is ready, buyer need to deploy the application as per the deployment steps provided in this guide.

Note: ECCA application infrastructure code will be available in public git repository as terraform stack. ECCA application will be available as image in AWS marketplace container registry. The ECCA infrastructure code will be accessible publicly. Although, ECCA application (container Image) will only be available to licensed buyers.

## Lead Time For Deployment

The application ECCA AWS is very light weight. The average time for deployment is between 40 - 45 minutes for the end-to-end deployment. The AWS Infra deployment will need approximately 20 to 25 min to deployment and application deployment takes around 20 minutes.

## Supported Regions

ECCA will be supported in all AWS regions where all the AWS Services which are mentioned in table 3 are supported. The regions where all AWS services are supported are listed in the URL (https://aws.amazon.com/about-aws/global-infrastructure/regional-product-services/)

## Technical Pre-requisites

* Customer needs to have an AWS Account and user configured with Administrator privileges. We recommend that root user is not used for solution deployment as prescribed by AWS best practices and is secured by MFA.
* A general knowledge of the GHG protocol, of emission factors and basic concepts of ESG are useful to correctly define the inputs and interpret the results.
* The customer needs to have Proper license to run ECCA application.
* The customer needs a S3 bucket provisioned to store the tfstate file of Terraform (This can be performed in deployment [Step 5](#_9.1_Step_by)).

## Technical Skills Required

* Linux or Windows CLI experience is helpful to run (Copy Paste) some command line scripts. Basic AWS understanding would be desirable. This does not require any scripting or programming skills.
* The customer should have basic understanding of deployments and Terraform basics.

## Environment Configuration

Terraform scripts deploys the all the required AWS services Infra in customer AWS account, it also deploys the application in same AWS account, Now infra parameters for the first time are to keyed in terraform plan command console, it saves the parameters in tf vars file.

The Infra parameters that are required are as mentioned below

1. aws\_region = "[aws region name **example:eu-west-2**]"
2. env = "[environment name **example:dev/prod**]"
3. cluster\_name = "[clustername **example:itco-cluster**]"
4. domain\_frontend = "[front end domain name. **Example**: ftend.ecca.com]"
5. domain\_backend = "[backend domain name **Example: bkend.ecca.com**]"
6. vpc\_name = "[name of the VPC]"
7. account\_id = "[aws account id]"
8. user\_email = "[user email id]"
9. psql\_instance\_name = "[SQL instance name]"

# Security

The ECCA application has used best practice approach to secure and protect client’s data. The security standards recommended by AWS were also taken into consideration for ECCA deployment on AWS platform. The ECCA application will run as service dependent on few tasks. The tasks get deployed in private subnet.

These services are deployed as individual containers. Each container has limited access to public network. All the containers are running process as non-root user.

The front application and backend application are on separate pods, if any pod goes down, other pod will be run without any dependency.

The application url is only accessible via https. Ex(<https://acca.xxxx.xxx.atos.net>)

## 7.1 Deployment Privileges

Application does not need AWS root privileges for deployment. We recommend customer to create an admin user with CLI access to deploy the Infra and application.

## IAM Roles and Policies

Application needs the below roles for Quicksight Embedding and integration.

|  |  |  |  |
| --- | --- | --- | --- |
| **Service** | **Access level** | **Resource** | **Request condition** |
| QuickSight | Limited: Write | Multiple | None |
| CloudWatch Logs | Limited: Write | All resources | None |

## 7.3 Authentication & Authorization

The ECCA application has in built user management. The user management will be handled by Super Admin account created at the time of deployment of ECCA. Data Admin will be able to upload inventories and taxonomy upgrades. General users will just be able to use the dashboard.

The AWS license manager will authorize the customer who have bought the application as per license plan, The authorized user will be able to install and run the application only in registered AWS account.

## Encryption

ECCA uses AWS managed encryption keys for PostgreSQL RDS, EKS and S3. So, data in PostgreSQL RDS, EKS and S3 is encrypted. The application supports encryption in Transit via SSL. However, the customer needs to bring their own certificates and configure the Elastic Load Balancer with a HTTPS listener.

Passwords are stored using spring BCryptPasswordEncoder.

## Secrets Management

Most of the Secrets will be stored in the AWS SSM parameters which would be delivered by next release of ECCA.

The user login credentials which are managed from application are stored in encrypted format in PostgreSQL.

## Customer Data

The customer sensitive data, inventory data after encryption is stored in same AWS RDS PostgreSQL instance. ECCA application encrypts the password and store it in AWS RDS PostgreSQL.

Refer to PostgreSQL backup

## Network Configuration

The network configuration will be provisioned by Terraform.

* Network configuration
  + 1 VPC is configured with a /16 CIDR range
  + 3 Public subnets configured with /27 CIDR range
  + 3 Private subnets configured with /27 CIDR range
  + 3 Private subnets for Database /27 CIDR range.
* Security Groups

The below Security groups are attached in ECCA application infrastructure.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| AWS Services which attached with security groups | Security Group Name | Inbound/Outbound | Port | protocol | Source | Comments |
| Lambda | Lambda-sg | Inbound | 443 | TCP | 0.0.0.0/0 | Allow HTTPS access to S3 from lambda |
| Inbound | 5432 | TCP | 0.0.0.0/0 | Allow access to RDS from lambda |
| Outbound | ALL | ALL | 0.0.0.0/0 | Allow outbound access to the internet |
| Postgre SQL | [env]-db-security | Inbound | 5432 | TCP | 10.0.1.0/27, 10.0.2.0/27, 10.0.3.0/27 | Allow access to RDS from different subnets |
| Outbound | 5432 | TCP | 10.0.1.0/27, 10.0.2.0/27, 10.0.3.0/27 | Allow access to different subnets from RDS |
| Cluster node | [clusterName] -node-[SGID] | Inbound | 8443/9443/1025/4443/10250/6443/443/53 | TCP |  | Cluster API to node [port]/tcp webhook |
| outbound | ALL | ALL | 0.0.0.0/0 | All traffic |
| Cluster | [cluster name]-cluster-[SGID] |  | 443 | https | [cluster name]-cluster-sgid | Node groups to cluster API |
| EKS | [eks name]-[SGID] | inbound | ALL | ALL | 0.0.0.0./0 | All traffic |
|  |  | outbound | ALL | ALL | 0.0.0.0./0 | All traffic |

# Cost

## 8.1Service Costing

The table below lists the minimum billable and non-billable AWS services to deploy ECCA on AWS. The table also represents the services deployed by user using ECCA Application. For example, when user creates a

NOTE: AWS subscription is owned by the customer and billed to the customer.

NOTE: Few AWS Services when deployed but not running may still incur charges. This has been indicated by column IDLE below in table.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Services | Billable if Idle | Mandatory | Note | For more details |
| AWS EKS | Checkmark with solid fill | Checkmark with solid fill | Billed based on usage, hourly charge is applicable. | <https://aws.amazon.com/eks/pricing/> |
| AWS QuickSight | Checkmark with solid fill | Checkmark with solid fill | Billed based on reader session. Note: each reader session is 30 minutes | <https://aws.amazon.com/quicksight/pricing/> |
| AWS CloudWatch |  | Checkmark with solid fill | Billed based on log storage, log delivery. | <https://aws.amazon.com/cloudwatch/pricing/> |
| AWS S3 |  | Checkmark with solid fill | Billed based on data storage, data retrieval, data transfer, data management, data replication, data query. | <https://aws.amazon.com/s3/pricing/> |
| AWS RDS | Checkmark with solid fill | Checkmark with solid fill | Billed based on usage, hourly charge is applicable | <https://aws.amazon.com/rds/pricing/> |
| AWS Elastic cache | Checkmark with solid fill | Checkmark with solid fill | Billed based usage, hourly charge is applicable | <https://aws.amazon.com/elasticache/pricing/> |
| AWS Secrets Manager |  | Checkmark with solid fill | Billed based on the number of secrets stored and API calls made | <https://aws.amazon.com/secrets-manager/pricing/> |
| AWS NAT Gateway | Checkmark with solid fill | Checkmark with solid fill | hourly charge is applicable | <https://aws.amazon.com/vpc/pricing/> |

## 8.2 Solution Costing

ECCA solution follows an annual subscription fee when sold to client directly. There will be charges for QuickSight Enterprise version licensing fee and AWS services usage additionally.

## 8.3 Solution Sizing

Solution size is described in the Terraform.

* Solution is based on EKS for Compute which is using EC2 as nodes and its auto scalable up to max three pods. At a time 2 nodes will be up & running. The compute requirements are t2.micro EC2 is CPU 1 vCPU (Virtual CPU) & Memory 1 GiB (Gibibyte) RAM
* The database RDS PostgreSQL is configured with Instance class db.t3. micro, vCPU2 & RAM 1GB is single node.

# Installation / Deployment

## 9.1 Step by step deployment

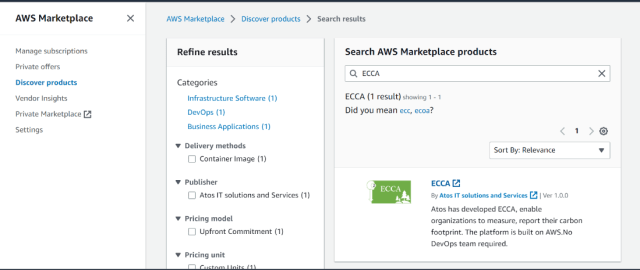
Process for subscribing AWS product from marketplace:

**Sign In to AWS Console:**

Log in to your AWS Management Console using your AWS account credentials.

**Navigate to AWS Marketplace:**

From the AWS Management Console dashboard, locate and click on the "AWS Marketplace" link. This might be under the "Services" section or accessible from the "Find Services" search bar.

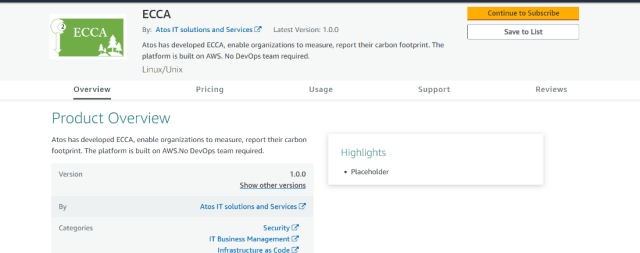


**Select the Product:**

Click on the product's name or thumbnail to view the product details page.

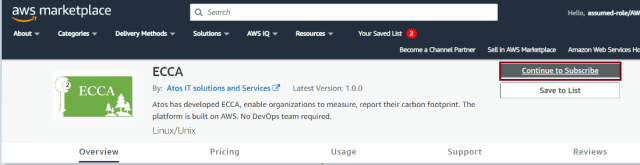
**Review Product Details:**

On the product details page, you'll find information about the product, its pricing, and other relevant details. Review this information to ensure the product meets your requirements.



**Subscribe to the Product:**

If you're ready to proceed, click the "Continue to Subscribe" button or any similar button that indicates subscription.

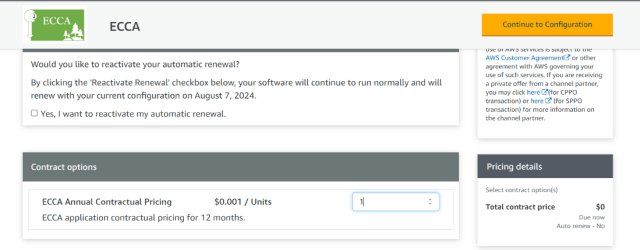


**Choose Subscription Plan:**

Some products might offer different subscription plans with varying features and pricing. Select the plan that suits your needs and click "Continue" or an equivalent button.

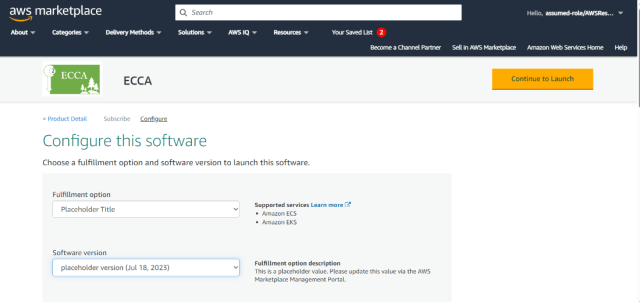
**Configure Product Settings:**

Depending on the product, you might need to configure certain settings before subscribing. This could include options like instance type, region, and other preferences.



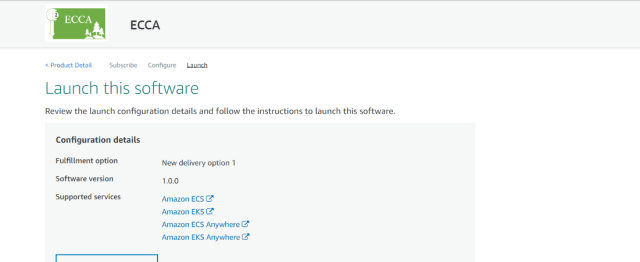
**Confirm Subscription:**

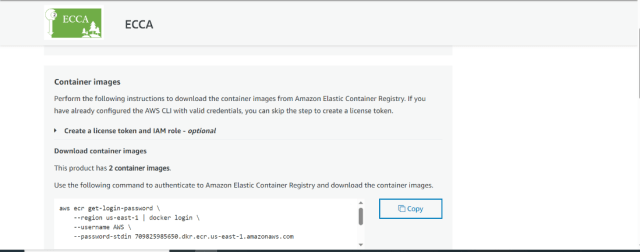
After providing the necessary information, confirm your subscription. You might need to click a "Launch" button.



**Wait for Subscription Activation:**

Once you've subscribed, the product might take some time to activate. During this time, the necessary resources and configurations are set up.





After the purchase of ECCA customer will receive

* User documentation
* Container images that contain the ECCA Software
* Link to download IAAC Terraform code to create infrastructure

Prerequisites:

1.Need IAM User as pre-requisite to perform all the deployment tasks. To create an IAM user, follow these steps

1. Sign into the AWS Management Console at https://console.aws.amazon.com/.
2. Open the IAM service by searching for "IAM" in the AWS Management Console search bar and selecting the "IAM" result.
3. In the left navigation pane, click on "Users" and then click on the "Add user" button.
4. Enter a username for the IAM user in the "User name" field.
5. Under "Access type", select the checkbox for "Programmatic access" to enable CLI access.
6. Under "Console password", choose one of the following options:
   1. "Autogenerated password": This option will automatically generate a secure password for the user.
   2. "Custom password": This option allows you to specify a custom password for the user.
7. (Optional) If you chose "Custom password" in the previous step, enter a custom password for the user in the "Custom password" field. Make sure to follow AWS password policy requirements.
8. Under "Permissions", select the checkbox for "Add user to group" and click on the "Create group" button.
9. In the "Create group" dialog, enter a group name for the IAM group. For example, you can enter "cli-group".
10. In the "Attach policies" step, select the checkbox for "AdministratorAccess". This policy provides complete admin access to the user.
11. Click on the "Create group" button to create the group and attach the policy.
12. Back in the "Add user" page, review the user's information, including the access type, console password settings, and group membership.
13. Click on the "Next: Tags" button if you want to add any tags to the user (optional). Otherwise, click on the "Next: Review" button.
14. Review the user's configuration details and click on the "Create user" button.
15. On the "Success" page, you will see the user's access key ID and secret access key. Make sure to download the CSV file or copy these credentials because you won't be able to access them again.

Need github client clone and execute the code. To install the git client, refer below link.

**https://git-scm.com/downloads**

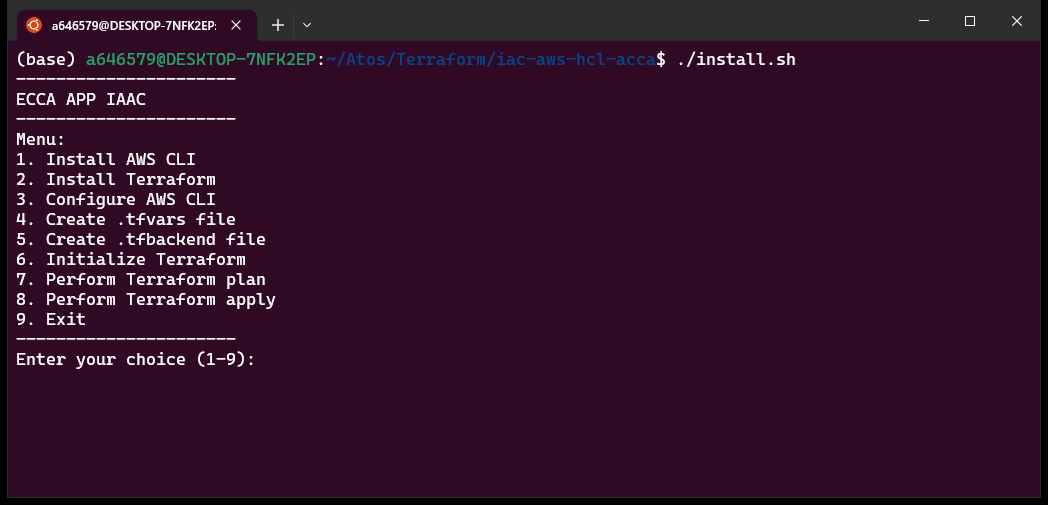
To generate AWS infrastructure, we are using Terraform (IAAC) language. To deploy the AWS infrastructure for application hosting, follow these steps in sequential order.

* Open a terminal or wsl.
* if your Terraform files are in the directory called “terraform”, navigate to the folder “terraform”
* Enter “git clone <github-iaac-repo>”
* Note: <github-iaac-repo> link will be provided in the deployment guide.
* Now you are find the repo <github-iaac-repo> under terraform directory.

We have streamlined the deployment process in one-go-deployable using shell script.

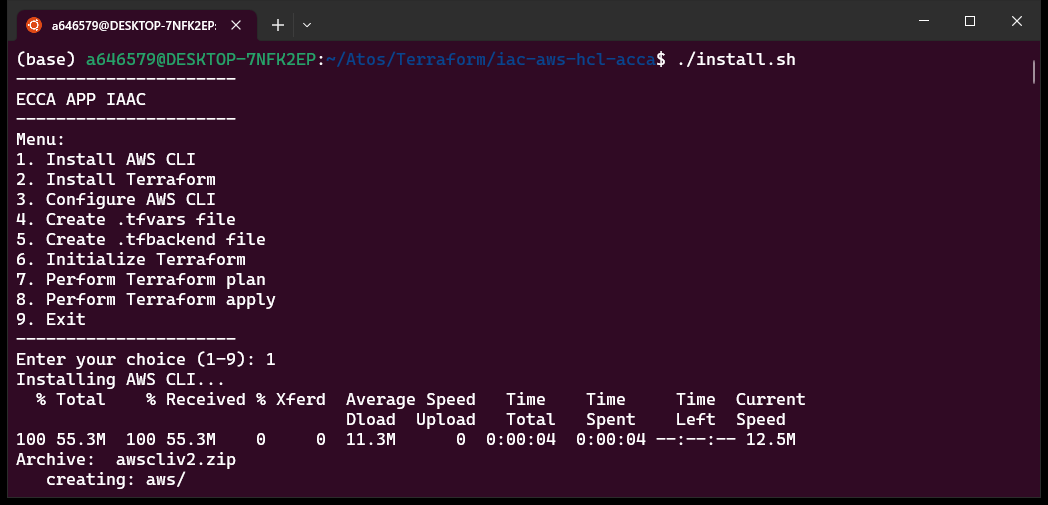
**Infrastructure Deployment:**

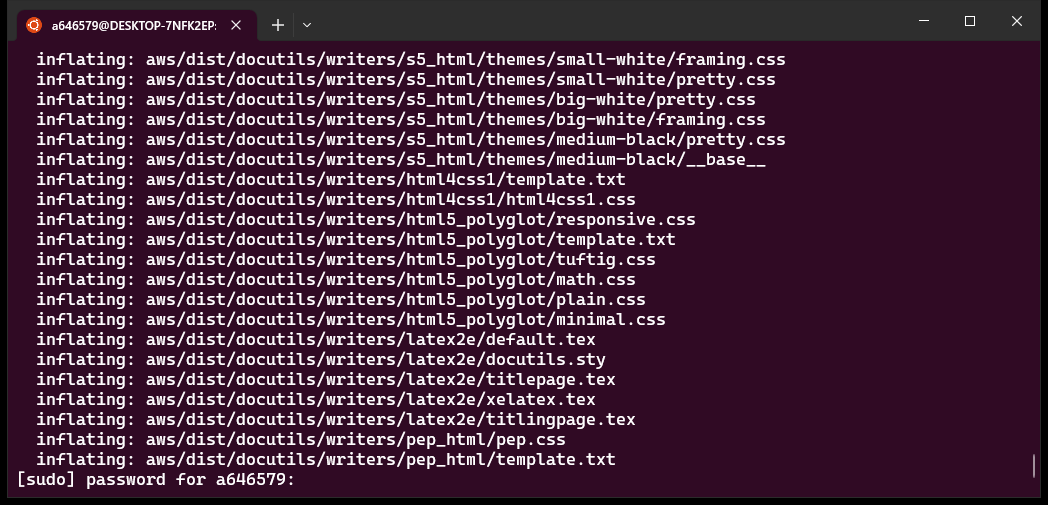
Run the ./install.sh script which is in “<github-iaac-repo>” directory. It will show the 9 steps. Each step has to be run sequentially.



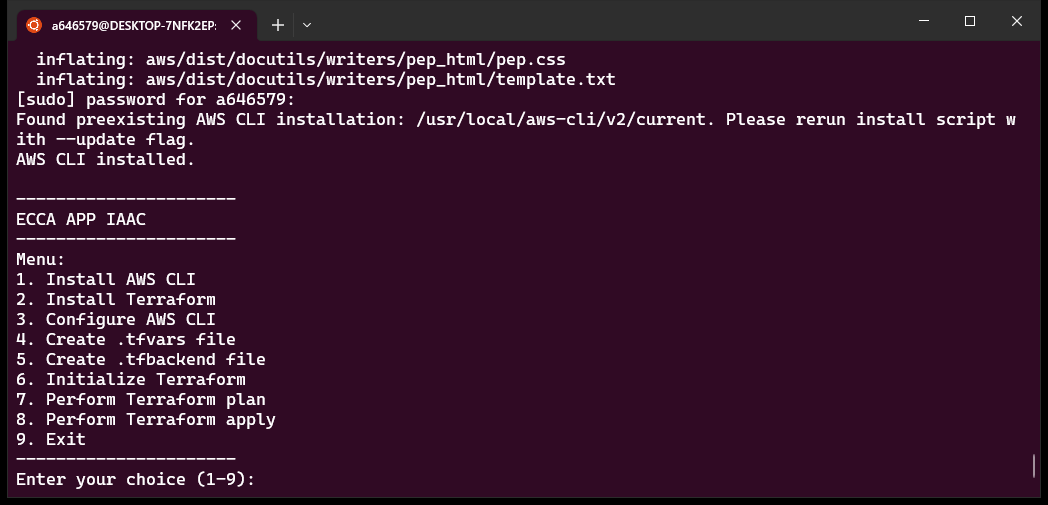
* If you have installed AWS CLI and terraform in your system, then skip 1-3 items, otherwise follow the below steps:

1. Install AWS CLI

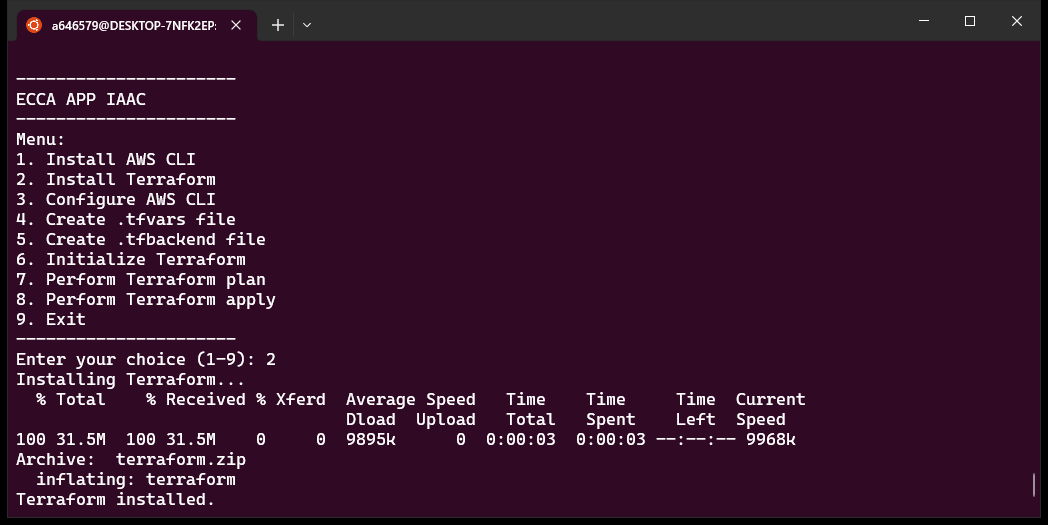




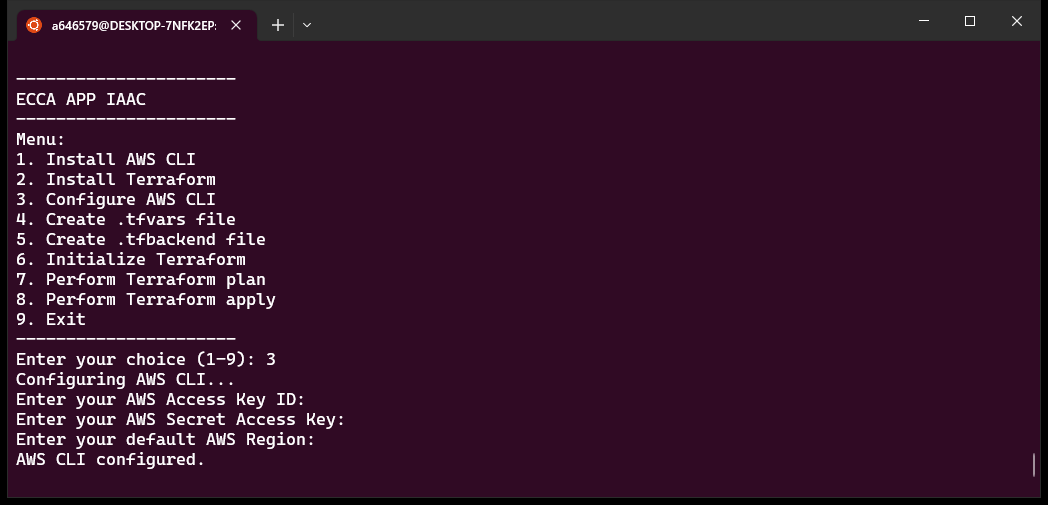
Enter the password to perform the installation. On completion you will see AWS CLI installed as below.



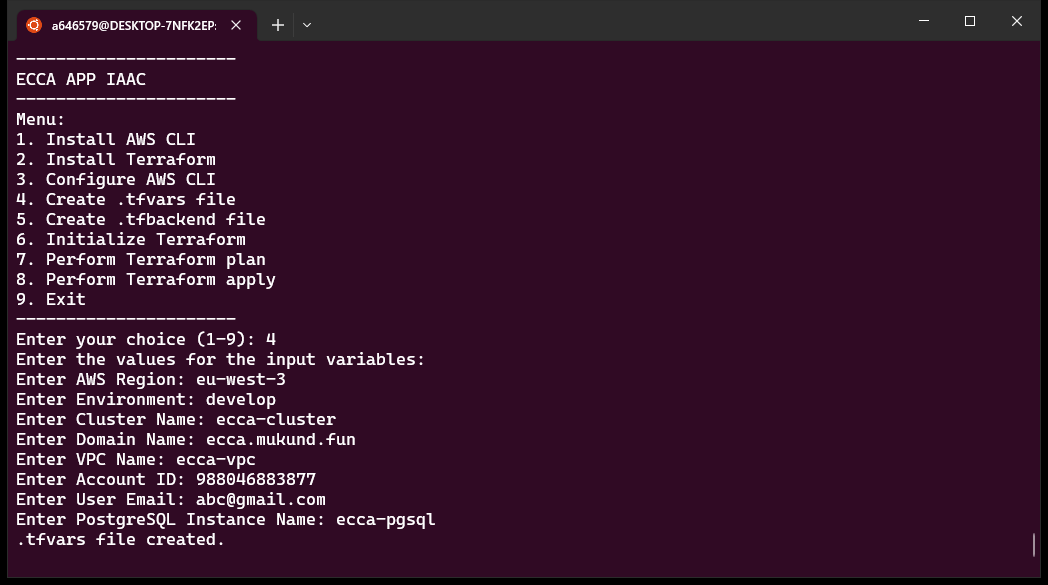
1. Install Terraform



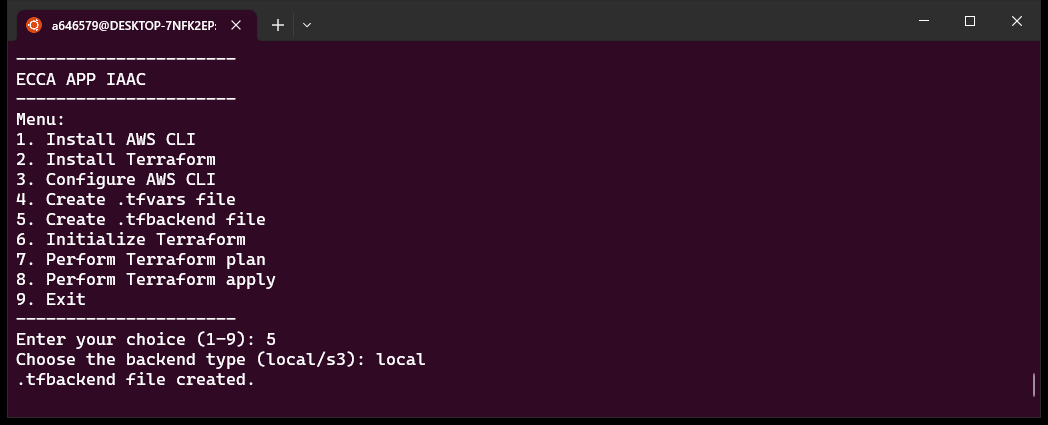
1. Configure AWS CLI. Enter the AWS Access key id, secret access key and Aws Region. If requested to enter the format, enter ‘json’



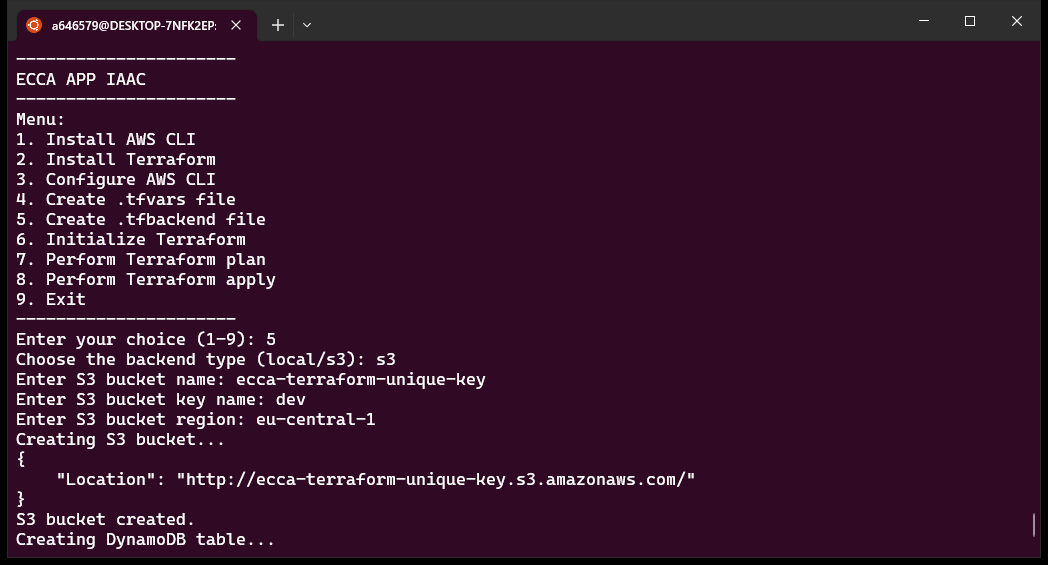
1. Create tfvars file. This file is required to add the variables as input for the application deployment. Enter the values as per the request.

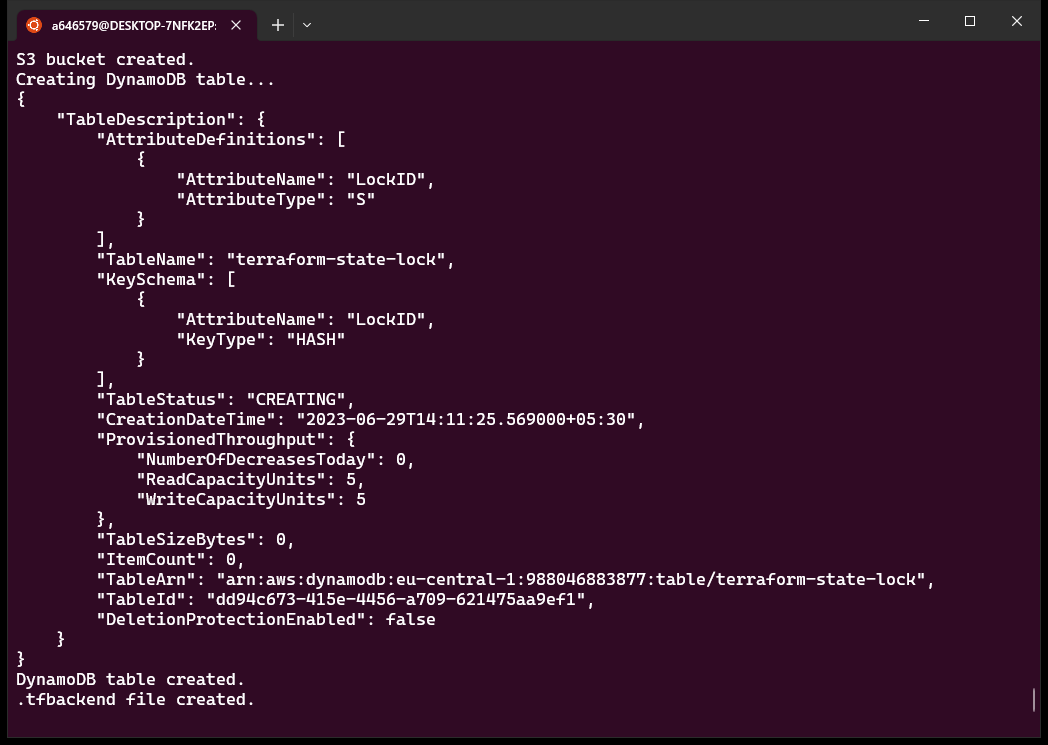


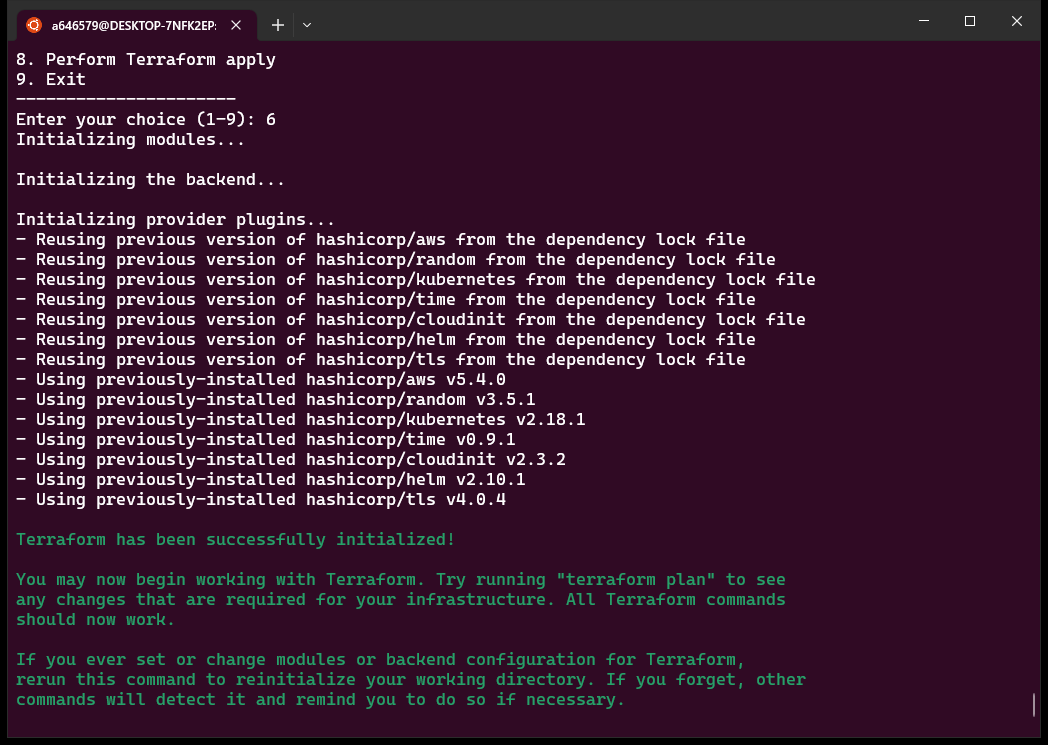
1. Create tfbackend file. This will ask you to select the location of tfstate file. You will get 2 options local and s3.
   1. If you are selecting local, please make sure to store the tfstate file somewhere safe.

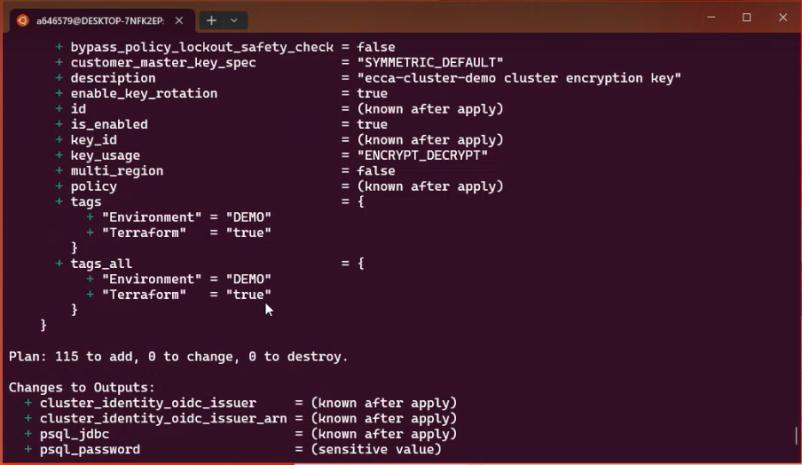


* 1. If you select s3, below options will be requested and depending on that the s3 bucket and dynamodb will be automatically created for you. Make sure to give unique name for s3 bucket.

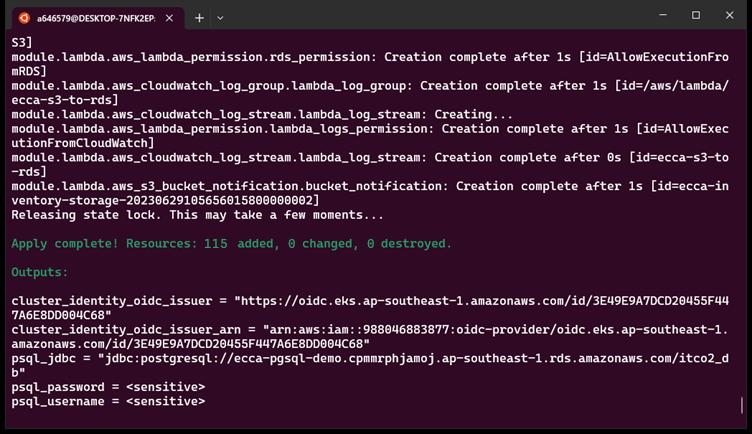




1. Terraform Initialize: this is to install the plugins to run the terraform code. 
2. Terraform Plan: Using this command the terraform will plan out of how many resources to be created or needs to be updated (if run second time). In case if you find any errors (ignoring the warnings), please contact support team.



1. Terraform Apply: Terraform apply to start the infrastructure creation. It will prompt you to enter yes if you are ok to go with the plan.

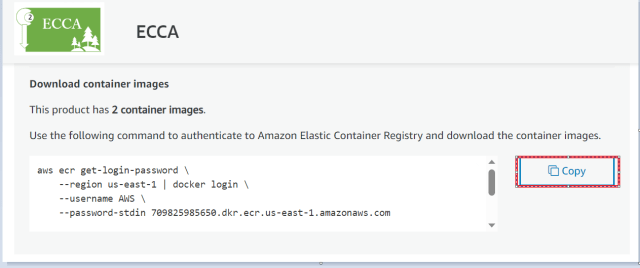


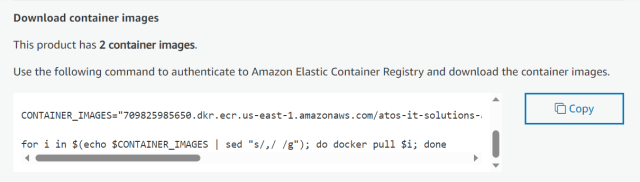
1. Now, navigate to postrun folder to perform pre-requisites resources on eks cluster.
2. Perform the similar steps, – Terraform init, plan and apply to deploy postrun scripts to EKS.
3. Infrastructure deployment will be completed after run all the steps.

Note:

1. Terraform init, plan and apply steps needs to be run sequentially, otherwise infrastructure will not be created.
2. If you have installed AWS CLI and terraform in your system, we can skip the steps 1-3 items.
3. We have created DNS Certificate for you in AWS Certificate Manager. You need to validate it before moving on to application deployment. Follow [Section 9.2](#_9.2_Post_app) to perform the same.

Once infrastructure is successfully created, user can proceed towards downloading the container image to the local. For that copy the commands from aws marketplace as shown below





**Command looks like this:**

aws ecr get-login-password \

--region us-east-1 | docker login \

--username AWS \

--password-stdin [AccountID].dkr.ecr.us-east-1.amazonaws.com

CONTAINER\_IMAGES="[AccountID].dkr.ecr.us-east-1.amazonaws.com/atos-it-solutions-and-services/ecca-frontend:1.0.0, [AccountID].dkr.ecr.us-east-1.amazonaws.com/atos-it-solutions-and-services/ecca-backend:1.0.0"

for i in $(echo $CONTAINER\_IMAGES | sed "s/,/ /g"); do docker pull $i; done

Note :

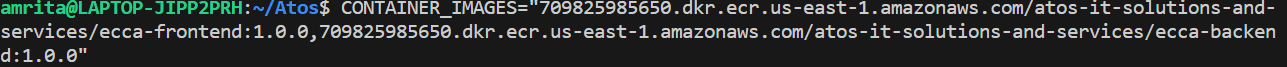
**Application frontend image** : [AccountID].dkr.ecr.us-east-1.amazonaws.com/atos-it-solutions-and-services/ecca-frontend:1.0.0

**Application backend image**: [AccountID].dkr.ecr.us-east-1.amazonaws.com/atos-it-solutions-and-services/ecca-backend:1.0.0

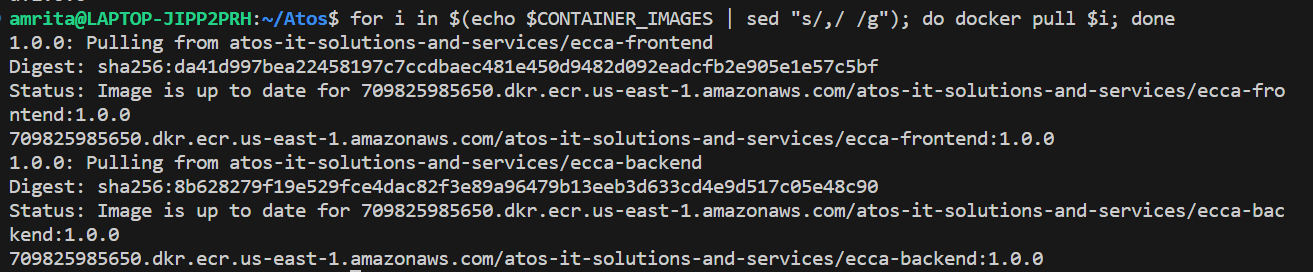
* Run the first command as shown in above picture. Output should look like below:



* Run the second command to set the container image. Output should look like below:

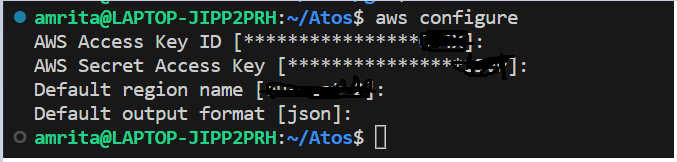


* Run the third command to download the application frontend & backend images to the local server. Output should look like below:

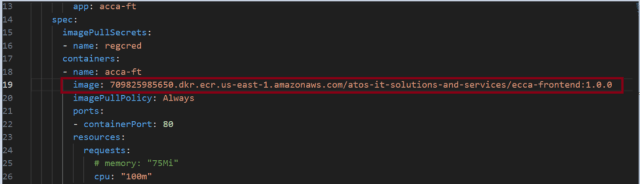


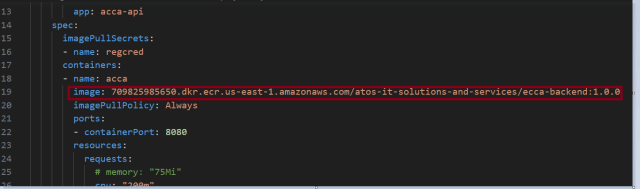
Once ECCA application images have been downloaded to local. User can update this frontend & backend images in the deployment file which is available in github repository (Link will be shared in deployment guide for application deployment)

* Before starting the deployment of application, do configure your access key & secret access key with desired region as per requirement:



* Once credentials has been set for required region ,User can update both frontend & backend images in the deployment yaml files as shown below:





* Before starting the deployment, update the config file by running below command:

Exp: aws eks update-kubeconfig --region eu-west-2 --name itco-cluster

itco-cluster-->EKS Cluster name

eu-west-2--->Required region, where infrastructure is running



* Once images are updated in the deployment yaml files, User can proceed with create namespace for frontend application deployment, as shown below:

**Command:** kubectl create namespace acca-phase2-ft



* Once frontend namespace has been created for frontend application deployment, create secret to pull application frontend image from docker registry.

**Command:** kubectl create secret docker-registry regcred –docker server=[AWSAccountID].dkr.ecr.eu-central-1.amazonaws.com --docker-username=AWS --docker-password=$(aws ecr get-login-password) --namespace=acca-phase2-ft



* Once secret created, let’s start with deployment of frontend application.

**Command:** kubectl apply -f deployment.yaml -n acca-phase2-ft



* After deploying the frontend application, user can check, if pods are up & running using below command.

**Command:** kubectl get po -n acca-phase2-ft



* Once deploying frontend is successful, User can proceed with deploying the service.yaml file to access the application.

**Command:** kubectl apply -f service.yaml -n acca-phase2-ft



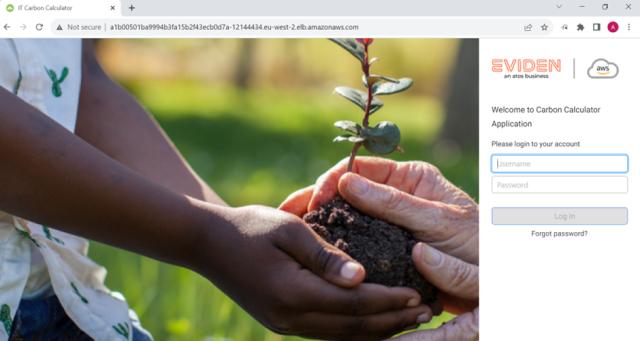
To check successful deployment of service.yaml file, User can run below command:

**Command**: kubectl get svc -n acca-phase2-ft



By running service.yaml file, one External-IP has been assigned to the pod to access application from the internet.

By running External-IP in browser ,User can able to access the application.



* Same like frontend deployment ,User need to deploy the backend as follows:
* Create backend namespace using below command.

**Command:** kubectl create namespace acca-phase2-bk



* Create secret for backend to pull image from docker registry.

Command: kubectl create secret docker-registry regcred --docker- server=988046883877.dkr.ecr.eu-central-1.amazonaws.com --docker-username=AWS --docker-password=$(aws ecr get-login-password) --namespace=acca-phase2-bk



* Once secret has been created,deploy the backend of the application as follows:

**Command:** kubectl apply -f deployment.yaml -n acca-phase2-bk



* Run below command to check deployment is successful.

**Command:** kubectl get po -n acca-phase2-bk



* Once deployment is successful, deploy the service.yaml file to make your application expose to internet.

**Command:** kubectl apply -f service.yaml -n acca-phase2-bk



Once service.yaml file has been deployed, User can run below command to check if service has been deployed successfully.

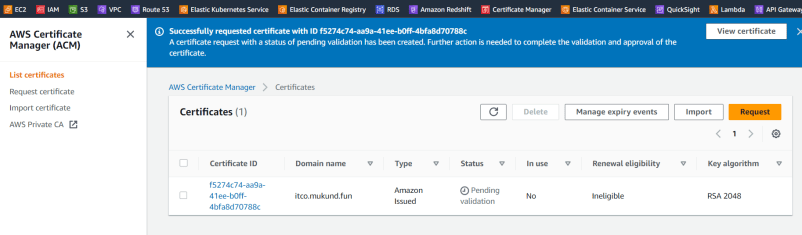
**Command:** kubectl get svc -n acca-phase2-bk



## 9.2 Post app installation (Validation of DNS with AWS SSL certificate)

To perform DNS validation for an ACM (AWS Certificate Manager) certificate, follow these steps:

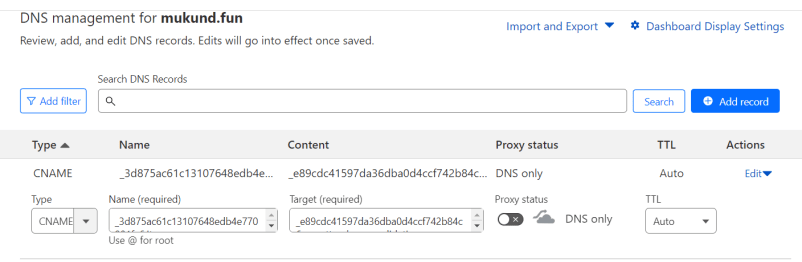
1. Open the AWS Management Console and navigate to the ACM service.
2. Select the region where your certificate is located.
3. Find the certificate for which was created via terraform to perform DNS validation and click on its ARN to open its details.



1. In the certificate details page, locate the "Validation" tab and click on the "Create record in DNS" button.
2. A dialog box will appear with instructions on how to manually create a DNS record. Take note of the "Domain name" and "Record value" provided in the instructions.
3. Open a DNS management interface for your domain provider or DNS hosting service. This could be the web interface of your domain registrar or a separate DNS management tool.
4. Create a new DNS record with the details provided in the instructions. The record type and value will be specified.

For example, if the instructions provide the following information:

* + Domain name: \_1234abcd.example.com
  + Record type: TXT
  + Record value: \_1234abcd.example.com.acm-validations.aws.



In this case, you would create a new TXT record with the name \_1234abcd.example.com and the value \_1234abcd.example.com.acm-validations.aws..

1. Wait for the DNS record to propagate, which can take some time depending on DNS propagation settings and TTL (Time to Live) values.
2. Return to the ACM console and refresh the certificate details page.
3. In the "Validation" tab, you should see the status of the validation change from "Pending validation" to "Success" once the DNS record has propagated and the validation is complete.
4. Once the validation is successful, the certificate is ready to be used in your AWS services, such as Elastic Load Balancer (ELB)

# 10.0 Troubleshooting

Deployment issues: Sometimes it might happen that terraform apply fails. You can raise case with Eviden Support to resolve the issue. You can collect the screenshot of failure error.

Few things, you need to check:

1. The s3 bucket name needs to be unique across the world. Hence while creating the backend please make sure to add unique name only.

**Health Checks for cluster & pods**

To check the health of an Amazon Elastic Kubernetes Service (EKS) cluster, you can use the following approaches:

AWS Management Console:  
a. Log in to the AWS Management Console.  
b. Navigate to the Amazon EKS service.  
c. Select your cluster from the list.  
d. In the "Cluster details" section, check the "Health" status. It should indicate whether the cluster is active, updating, or has encountered any issues.

AWS CLI:  
a. Install and configure the AWS CLI if you haven't already.  
b. Open a terminal or command prompt.  
c. Run the following command to check the cluster status:

 aws eks describe-cluster --name <cluster-name> --region <region>  
Replace <cluster-name> with the name of your EKS cluster and <region> with the AWS region where your cluster is located.  
d. Look for the "status" field in the command output. It should indicate the health status of the cluster.

Kubernetes CLI (kubectl):  
a. Install and configure the Kubernetes CLI (kubectl) if you haven't already.  
b. Open a terminal or command prompt.  
c. Run the following command to check the status of the cluster components:

 kubectl get componentstatuses

 d. Examine the output to see the status of various cluster components like the control plane, etcd, and others. A healthy cluster will show "Healthy" for all components.

 Cluster-autoscaler:  
If you have enabled the cluster autoscaler, you can check its logs or metrics to ensure it is functioning correctly and scaling the cluster nodes appropriately based on the workload demands.

To check health checks on pod level you can follow below commands:

kubectl get po -n <namespace>

 Replace <namespace> with the specific namespace where your pods are deployed. If you're not using namespaces, you can omit the -n <namespace> part.  
d. Examine the output to see the status of each pod. The status column will indicate if a pod is running, completed, or has encountered any issues.  
e. You can also use additional flags with the kubectl get pods command, such as -o wide for more detailed information or --watch to continuously monitor the pod status in real-time

**POD Status Check**

user needs to update config file with required region & eks cluster in aws cli then

run kubectl command to see the pod details we can describe it

Currently we are using PostgreSQL monitoring via sending Cloud Watch logs.

**Handling faulty Conditions:**

1. Access Denied: This error typically occurs when the IAM user or role used by Terraform lacks the necessary permissions to create or modify AWS resources.
   * Ensure that the IAM user or role associated with your Terraform configuration has the required permissions. Refer to the AWS documentation or use predefined AWS IAM policies like "AmazonEC2FullAccess" or "AmazonS3FullAccess" to grant the necessary privileges.
   * Verify the access key and secret key credentials for the IAM user or role being used by Terraform.
2. Resource Already Exists: This error suggests that the resource you are trying to create with Terraform already exists in the target AWS account.
   * Check if the resource already exists in your AWS account. You can search for it manually in the AWS Management Console or use the AWS CLI.
   * If the resource was manually created outside of Terraform, consider importing it into your Terraform state using the terraform import command. This allows Terraform to manage the existing resource.
3. InvalidParameterValue: This error occurs when one or more input parameters provided to an AWS resource are incorrect or not in the expected format.
   * Double-check the input values provided in your Terraform configuration. Verify that they match the required format and are valid for the specific resource you're trying to create.
   * Consult the AWS documentation for the resource you're working with to ensure you're providing the correct values for its configuration.
4. Insufficient Permissions: This error indicates that the IAM user or role used by Terraform has some permissions, but they are not sufficient to complete the requested operation.
   * Review the required permissions for the operation that triggered the error. Update the IAM user or role policies accordingly to grant the necessary permissions.
   * Make sure to include any additional policies or permissions required by AWS services or resources referenced in your Terraform configuration.
5. Rate Exceeded: This error occurs when you exceed the service limits or quotas defined by AWS.
   * Check the service limits for the specific AWS service you're using. You can find the limits in the AWS Management Console or refer to the AWS documentation.
   * Request a limit increase if necessary. AWS provides a process to request higher limits for various services.

**Step to recover the software:**

If any error/crash occurs in ecca application, connect with support team based on the support plan.

Pods in Kubernetes can encounter below errors:

ImagePullBackOff: This error occurs when the pod fails to pull the required container image. It may happen due to issues with image availability, authentication, network connectivity, or incorrect image name or tag.

CrashLoopBackOff: If a pod repeatedly crashes and fails to start, it enters a CrashLoopBackOff state. This error can occur due to application or container configuration issues, such as a misconfigured command, missing dependencies, or an error in the application code.

To resolve the ImagePullBackOff error in Kubernetes, you can follow these steps:

Verify Image Availability: Ensure that the container image you are trying to pull exists in the container registry specified in the pod configuration. Double-check the image name, including the repository and tag. If the image doesn't exist or the tag is incorrect, you will encounter an ImagePullBackOff error.

Check Authentication: If the container registry requires authentication, double-check the credentials used for pulling the image. Ensure that the username, password, or access token is correct. If necessary, regenerate the credentials and update the pod configuration with the correct authentication details.

Network Connectivity: Verify that the Kubernetes cluster has proper network connectivity to reach the container registry. Ensure that there are no firewall rules, network restrictions, or proxy configurations blocking the communication between the cluster nodes and the registry. Test the network connectivity using tools like ping or curl.

Secret Configuration: If you are using a private container registry and need authentication, ensure that the appropriate secret is configured in Kubernetes. Create a secret containing the registry credentials and mount it in the pod configuration to allow Kubernetes to authenticate and pull the image.

ImagePullSecrets: If you are using Kubernetes Service Account credentials to authenticate with the container registry, ensure that the correct imagePullSecrets are specified in the pod's configuration. These secrets allow the pod to authenticate and pull images from the private registry.

Check Cluster Nodes: Verify that the nodes in your Kubernetes cluster have the necessary container runtime (e.g., Docker) installed and properly configured. Ensure that the container runtime is running, has network connectivity, and can access the container registry.

Docker Configuration: If you are using Docker as the container runtime, ensure that it is configured correctly. Verify that Docker is running, has the necessary storage drivers configured, and can pull images from the container registry without any issues.

Resource Constraints: Check if the node where the pod is scheduled has sufficient resources (CPU, memory, storage) available to run the pod and pull the image. If the node is resource-constrained, the image pull may fail. You can try allocating more resources to the node or consider using a different node with sufficient resources.

Retry and Debug: After making any necessary changes or checks, retry the deployment of the pod. Monitor the pod's logs and events for any specific error messages or warnings that can help in troubleshooting. Look for any new error messages that may provide more insight into the ImagePullBackOff issue.

## 10.1 Scaling

* ECCA Core is configured for HPA (Horizontal pod autoscaling) with upper CPU limits as 100 m and threshold has been set to 80 % if CPU utilization hits the threshold autoscaling will be enabled. So minimum replica is set as 1 and maximum replica is set as 3.
* However, it will scale automatically based on requirements. We can modify the upper limits
* Autoscaling has been enabled both at cluster and pod level in EKS.

# Support

## 10.3 Scope

Basic Support Package:

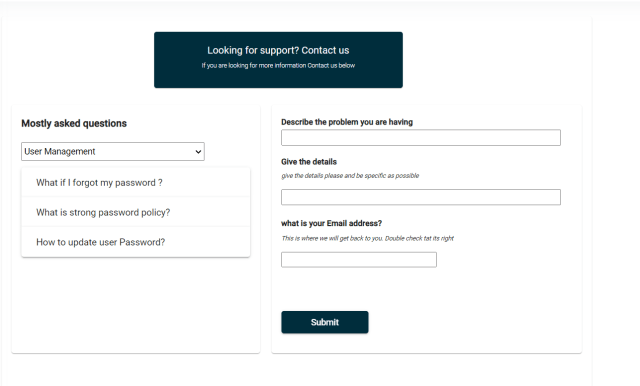
* Bug fix support
* Support staff is available during CET office hours (9 am to 5 pm) from Mon-Fri and is accessed via an online ticket platform.

## 10.4 Support duration

Support Duration will be subscription period of ECCA application.

## Support Plan

Typical support hours will be a working business day. Support will be provided by ECCA support team. Support staff is available during CET office hours from Mon-Fri and is accessed via an online ticket platform. Bug fix will also be provided via a ticketing system.



## Contact

Customer will be able to send an email to [dl-nzt-support@atos.net](mailto:dl-nzt-support@atos.net) customer support.

# Backup & Recovery

Weekly Backup is enabled for ECCA application stored in PostgreSQL.

# Routine Maintenance

## 11.1 Key & Credentials Change

ECCA uses JWT Secret Keys which exist for lifetime and doesn’t expire.

## Software Updates

The new image will be released on AWS marketplace repository from which customer can pull the latest update. This may also include new releases.

## License

ECCA is an annual subscription model. Addition to ECCA license, buyer should have QuickSight enterprise edition subscription and session per pricing monthly subscription.

## Pricing Models:

ECCA application is coming with flexible pricing model. Currently it supports “Bronze” model and details mentioned below. Additional pricings models will be added later.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Cost model Type | One-time base setup charges | Yearly subscription | Maximum number of users | Yearly IT asset limit | Data Retention period |
| ECCA | 1000 EURO per customer | 18000 Euro/year | Unlimited Users | 120000 IT assets inventory records | 1 year |

While buyer subscribes the products with any of the license model, the same license with entitlement detail will be replicated in Buyer’s AWS account license manager console.

If buyer’s AWS account is a member of an organization, they can share the license with the other accounts in their organization, but the buyer must set up license support in AWS Marketplace, and then share this from within AWS License Manager

ECCA application manages buyers license through API query. If buyers do not adhere to any component of entitlement of the license or exceed the limit, ECCA application will stop.

**Important Notes:**

AWS does not support asset limit control for granted license.

If buyer cancel the ECCA application subscription without terminating all running instances of the software, they will continue to be charged for any software usage.

## Service Limits

Customer must manage the AWS service limits. We recommend customers to use new account rather than existing account. This will avoid limit clash with existing AWS services. They can use the AWS Trusted Advisor service to monitor the usage against limits.

12.0 Glossary of Terms, Abbreviations and Acronyms

| Term | Meaning |
| --- | --- |
| ACM | AWS Certificate Manager |
| AD | Active Directory |
| ALB | Application Load Balancer |
| AMI | Amazon Machine Image |
| API | Application Programming Interface |
| ARN | Amazon Resource Name |
| AWS | Amazon Web Services |
| AZ | Availability Zone |
| BUP | Backup Account |
| CMDB | configuration management database |
| CI/CD | Continuous Integration / Continuous Development |
| CIDR | Classless Inter Domain Routing |
| CloudHSM | Cloud Hardware Security Module |
| CRM | Customer Relationship Manager |
| DB | Database |
| DNS | Domain Name System |
| DR | Disaster Recovery |
| DX | Direct Connect |
| EBS | Elastic Block Store |
| ECCA | Eviden Carbon Calculator |
| EC2 | Elastic Cloud Compute |
| EFS | Elastic File Store |
| EIP | Elastic IP. |
| ELA | Enterprise License Agreement |
| FMO | Future Mode of Operation |
| HLD | High Level Design |
| HTTPS | Hypertext Transfer Protocol Secure |
| IaaS | Infrastructure as a Service |
| IAM | Identity and Access Management |
| IP | Internet Protocol |
| KMS | Key Management Service |
| KPI | Key Performance Indicator |
| MFA | Multi-factor Authentication |
| NACL | Network Access Control List |
| NAT | Network Address Translation |
| OS | Operating System |
| OU | Organisational Unit |
| RDS | Relational Database Service |
| RI | Reserved Instance |
| RPO | Recovery Point Objective |
| RTO | Recovery Time Objective |
| S3 | Simple Storage Service |
| SDK | Software Development Kit |
| SG | Security Group |
| SLA | Service Level Agreement |
| SSH | Secure Shell |
| SSL | Secure Sockets Layer |
| TCP | Transmission Control Protocol |
| TGW | Transit Gateway |
| URL | Uniform Resource Locator |
| VPC | Virtual Private Cloud |
| VPN | Virtual Private Network |
| WAF | Web Application Firewall |
| EKS | Elastic Kubernetes Service |
| ACCA | Atos Carbon Calculator |
| MVP | Minimum viable product |
| JWT | JSON Web Tokens |
| TBD | To Be Decided |
| SAP | Systems, Applications, and Products |
| JSP | Java Server Pages |
| ETL | Extract Transform Load |
| AI/ML | Artificial Intelligence (AI) and Machine Learning |

Sample Inventory and Taxonomy excel sheets

