RBAC Module Usage Guide

Version 1.8

Table of Contents

Introduction

Welcome to **BAIDEAC's RBAC** (Role-Based Access Control) **Module**. This service is built as an executable for Ubuntu machines, featuring a Node.js server that supplies HTML components and securely manages user permissions through role-based access control. RBAC is a method of managing permissions by assigning roles to users, simplifying access management and improving security by limiting actions based on roles. This approach enhances security compliance and reduces administrative overhead, making it a powerful tool for handling access in applications with multiple user levels. In this guide, we'll walk you through getting started with BAIDEAC's RBAC Module.

Getting Started

Below, we'll explain how to set up the RBAC Module through AWS ECR download.

Pre-requisites

Before running the commands, ensure the following tools are installed and configured:

1. Docker

- Install Docker following the official instructions:
    - Docker for Windows
    - Docker for macOS
    - Docker for Ubuntu

Verify Docker installation:

docker --version

2. AWS CLI

- Install AWS CLI using the official documentation:

○ AWS CLI Installation

Verify AWS CLI installation:

aws --version

AWS Configuration

Before using AWS ECR, you need to configure AWS CLI with your credentials.

Run the following command to configure AWS CLI :
aws configure

1. Provide the following information when prompted :
    - **AWS Access Key ID**: Your AWS access key.
    - **AWS Secret Access Key**: Your AWS secret key.
    - **Default region**: `us-east-1` (or your desired region).
    - **Default output format**: `json` (or your preferred format).

3. License Key

You can get your license to use the platform from https://www.baideac.com/licensing

1. Create account
2. Request Trial Key
    - Select “RBAC”
    - Click on Request.
3. In the listed table, you can get the licence key.

4. Docker image as tar (Optional)

You can get your Docker image tar file from the licence page itself. Within the same license record, you can see a download button where you get the “rbac-docker-image.tar” file.

- Collect “rbac-docker-image.tar”

Getting Started with docker

With AWS Marketplace

1. Subscribe to a product on AWS Marketplace
2. You will see product details on the portal for the ECR product, repository information, etc. You will need the Docker repository information of the product. It should look something like this :

   ```
   709825985650.dkr.ecr.us-east-1.amazonaws.com/bhojr/baideac/rbac-container:1.8
   ```

3. Using the commands given from the AWS ECR portal, run the following:

   ```
   docker pull \
   709825985650.dkr.ecr.us-east-1.amazonaws.com/bhojr/baideac/rbac-container:1.8
   ```

With Docker image as tar

1. Go to the Download path where you downloaded "rbac-docker-image.tar"
2. Using the command given, you will be able to load image in your machine

```
docker load -i rbac-docker-image.tar
```

Run the Docker image

4. Then run the following command to build the Docker image :
```
docker run -it --name rbac-container \
-p 55501:55501 -p 55502:55502 \
-v pm2-data:/root/.pm2 \
709825985650.dkr.ecr.us-east-1.amazonaws.com/bhojr/baideac/rbac-container:1.8
```

Please refer to the two port numbers mentioned above (55501 and 55502). These represent the external and internal ports assigned to the frontend and backend. The port on the left (55501) corresponds to the **external** port, while the port on the right (55502) corresponds to the **internal** port. Keep in mind that the **internal ports** are configurable during the installation process, as shown in the below screenshots. On the other hand, the **external ports** are accessible for the frontend and backend.

5. After finishing the installation portion below, you can optionally download the preconfigured MongoDB if you wish to have a DB to connect the RBAC to. Run the following command from your machine that you installed the rbac-container in:

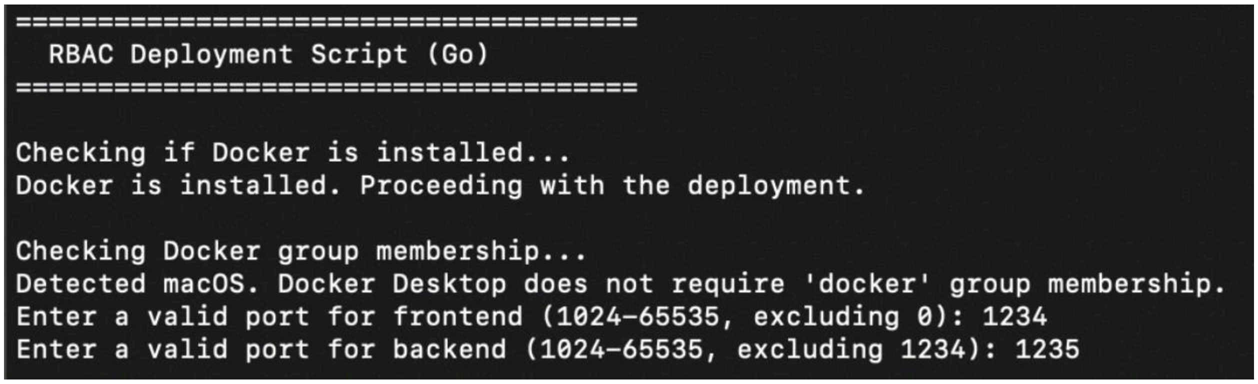
```
docker run -d --name rbac-mongo-db \
-p 55503:27017 \
-v rbac_mongodb_data:/data/db \
-v rbac_mongo_config:/data/configdb \
-e MONGO_INITDB_ROOT_USERNAME=admin \
-e MONGO_INITDB_ROOT_PASSWORD=admin \
mongo:6.0 --auth --port 27017 --bind_ip 0.0.0.0
```

Note the port numbers (`55503:27017`). You can change the EXTERNAL left hand side port number to whatever you wish. However, the INTERNAL right hand side port number MUST be 27017. After this step, your mongo uri will be:
`mongodb://admin:admin@<INTERNAL/PRIVATE_IP>:55503/admin`
Please note to use the internal/private IP for the mongo DB uri and not local host!!

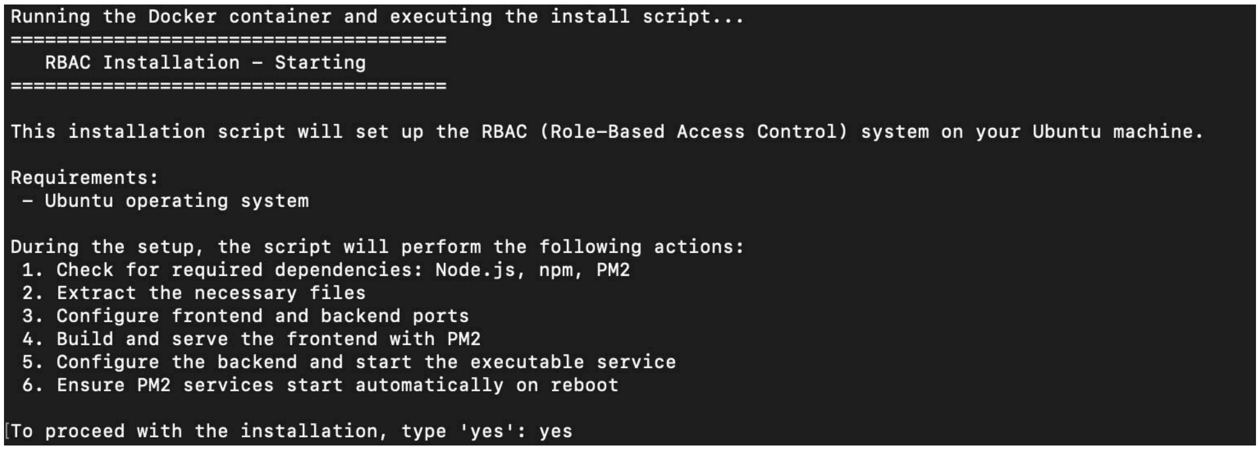
Installation Process

You will need to provide the following inputs when prompted:

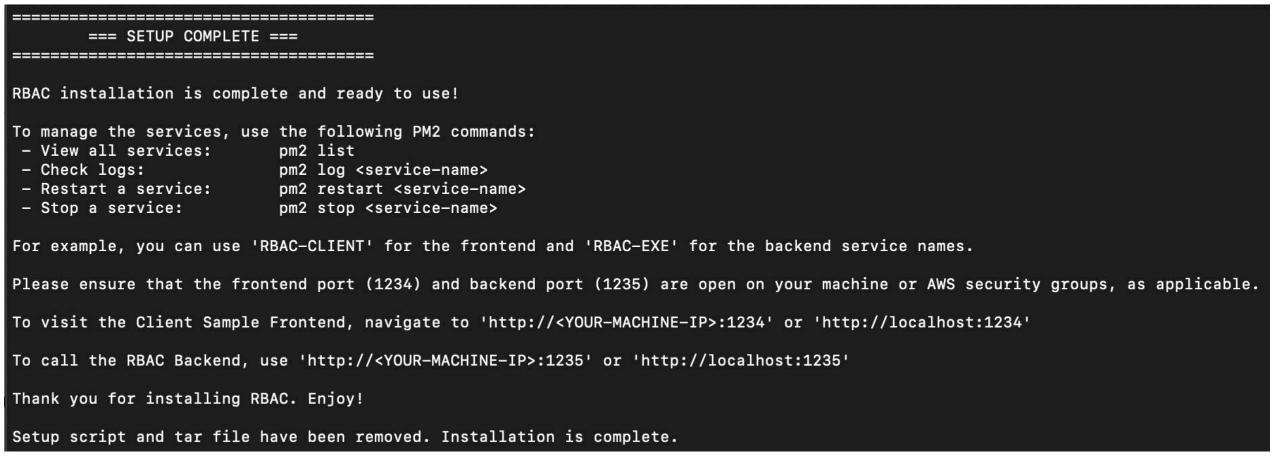
a. **Frontend Port**: This is a port to access the sample client repo for quick testing of the product.
b. **Backend Port**: This is a port to access the API calls in which your client app would be calling to use the RBAC service.



```
==========================================
  RBAC Deployment Script (Go)
==========================================

Checking if Docker is installed...
Docker is installed. Proceeding with the deployment.

Checking Docker group membership...
Detected macOS. Docker Desktop does not require 'docker' group membership.
Enter a valid port for frontend (1024-65535, excluding 0): 1234
Enter a valid port for backend (1024-65535, excluding 1234): 1235
```



```
Running the Docker container and executing the install script...
======================================
   RBAC Installation - Starting
======================================

This installation script will set up the RBAC (Role-Based Access Control) system on your Ubuntu machine.

Requirements:
 - Ubuntu operating system

During the setup, the script will perform the following actions:
 1. Check for required dependencies: Node.js, npm, PM2
 2. Extract the necessary files
 3. Configure frontend and backend ports
 4. Build and serve the frontend with PM2
 5. Configure the backend and start the executable service
 6. Ensure PM2 services start automatically on reboot

To proceed with the installation, type 'yes': yes
```

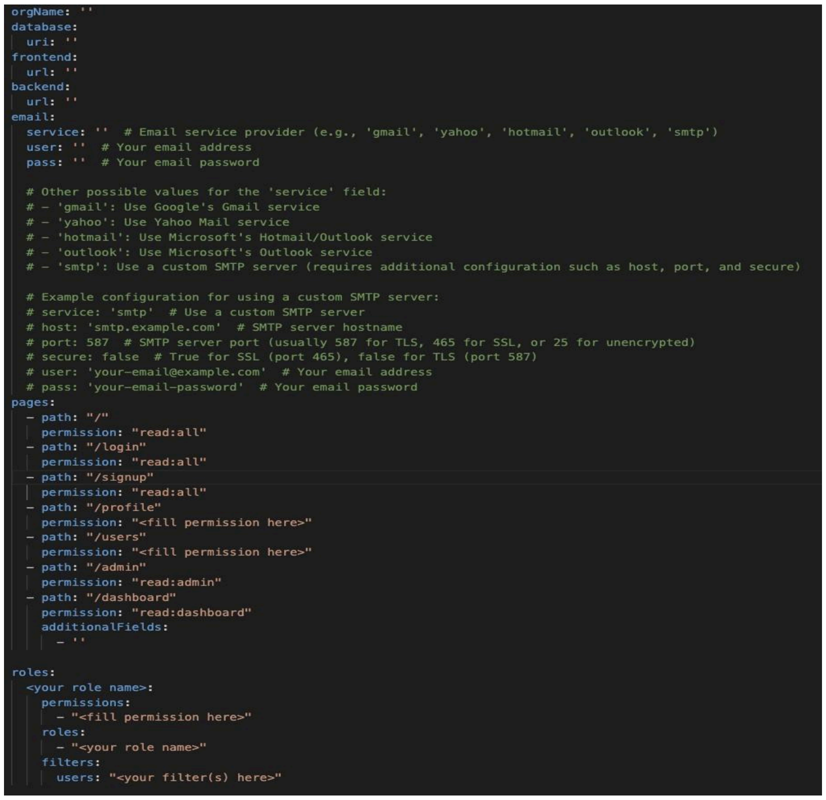


```
======================================
        === SETUP COMPLETE ===
======================================

RBAC installation is complete and ready to use!

To manage the services, use the following PM2 commands:
 - View all services:       pm2 list
 - Check logs:              pm2 log <service-name>
 - Restart a service:       pm2 restart <service-name>
 - Stop a service:          pm2 stop <service-name>

For example, you can use 'RBAC-CLIENT' for the frontend and 'RBAC-EXE' for the backend service names.

Please ensure that the frontend port (1234) and backend port (1235) are open on your machine or AWS security groups, as applicable.

To visit the Client Sample Frontend, navigate to 'http://<YOUR-MACHINE-IP>:1234' or 'http://localhost:1234'

To call the RBAC Backend, use 'http://<YOUR-MACHINE-IP>:1235' or 'http://localhost:1235'

Thank you for installing RBAC. Enjoy!

Setup script and tar file have been removed. Installation is complete.
```

These steps should have you set up quickly, ready to manage access with BAIDEAC’s RBAC Module.

Configuring the RBAC Module

After completing the initial setup, the next step is configuring the RBAC Module via the `**rbac.yaml**` file. This configuration file, located in the `**/app/core/etc**` directory, defines critical settings such as organisation name, database URI, frontend and backend URLs, and access permissions. Follow these steps to set up the `**rbac.yaml**` file:



```yaml
orgName: ''
database:
  uri: ''
frontend:
  url: ''
backend:
  url: ''
email:
  service: ''  # Email service provider (e.g., 'gmail', 'yahoo', 'hotmail', 'outlook', 'smtp')
  user: ''  # Your email address
  pass: ''  # Your email password

  # Other possible values for the 'service' field:
  # - 'gmail': Use Google's Gmail service
  # - 'yahoo': Use Yahoo Mail service
  # - 'hotmail': Use Microsoft's Hotmail/Outlook service
  # - 'outlook': Use Microsoft's Outlook service
  # - 'smtp': Use a custom SMTP server (requires additional configuration such as host, port, and secure)

  # Example configuration for using a custom SMTP server:
  # service: 'smtp'  # Use a custom SMTP server
  # host: 'smtp.example.com'  # SMTP server hostname
  # port: 587  # SMTP server port (usually 587 for TLS, 465 for SSL, or 25 for unencrypted)
  # secure: false  # True for SSL (port 465), false for TLS (port 587)
  # user: 'your-email@example.com'  # Your email address
  # pass: 'your-email-password'  # Your email password
pages:
  - path: "/"
    permission: "read:all"
  - path: "/login"
    permission: "read:all"
  - path: "/signup"
    permission: "read:all"
  - path: "/profile"
    permission: "<fill permission here>"
  - path: "/users"
    permission: "<fill permission here>"
  - path: "/admin"
    permission: "read:admin"
  - path: "/dashboard"
    permission: "read:dashboard"
    additionalFields:
      - ''

roles:
  <your role name>:
    permissions:
      - "<fill permission here>"
    roles:
      - "<your role name>"
    filters:
      users: "<your filter(s) here>"
```

1. **Open the File**: Navigate to the `**rbac-service**` directory and open the `**rbac.yaml**` file. This file includes essential fields that need to be inputted to align with your application requirements.

2. **Edit Configuration**: `**orgName**`: Fill in your organization or service name. This name will be displayed across the RBAC Module, representing your website.
   orgName: 'YOUR ORG NAME HERE'

3. **Edit Configuration: `database`:** Provide the URI of your MongoDB database. RBAC comes preinstalled with a MONGODB. The port for your MongoDB was configured in the installation step. The URI will be as follows:
   database:
   uri: 'mongodb://admin:admin@<PRIVATE_IP>:<MONGO_PORT>'

4. **Edit Configuration: `backend url`:** Set this URL with your EC2's public DNS as well as the backend port in the installation to allow frontend components to access backend APIs.
(i.e. *http://ec2-12-34-567-89.compute-1.amazonaws.com:3000*)

   If you are configuring the RBAC module on a separate machine, please use the hostname of your service and the backend port assigned during the installation process.

```
backend:
    url: 'http://<your-ec2-public-ipv4-dns>:BACKEND_PORT
```

5. **Edit Configuration: `frontend url`:** Enter the URL for your application's frontend. This URL is used by the RBAC Module to navigate back to the home page after a password reset. Set this URL with your EC2's public DNS as well as the frontend port.
(i.e. *http://12.34.567.89:3001*)

   If you are configuring the RBAC module on a separate machine, please use the hostname of your service and the frontend port assigned during the installation process.

```
frontend:
    url: 'http://<your-frontend-url>:FRONTEND_PORT
```

6. **Edit Configuration: `email`:** Input the email settings you want to use for account-related notifications. For security, we recommend using an app password or passkey.

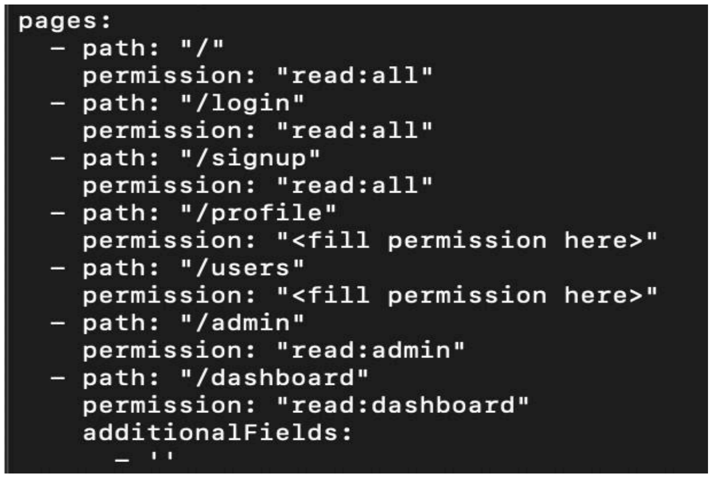
```
email:
    service: '' # Email service provider (e.g., 'gmail', 'yahoo', 'smtp')
    user: '' # Your email address
    pass: '' # Your email password
```

7. **Edit Configuration: `pages`:** In the `pages` section, you'll find the available routes, each with permissions and paths.

   Define permissions using the `read:` prefix, followed by the role with the lowest hierarchy that should have access (e.g., `read:user`).

   You can add more custom paths by following this structure. Ensure the permissions align with your application's access control requirements.

   For the dashboard route, you can add additionalFields for additional/custom DB fields you would want displayed in the dashboard.



```
pages:
  - path: "/"
    permission: "read:all"
  - path: "/login"
    permission: "read:all"
  - path: "/signup"
    permission: "read:all"
  - path: "/profile"
    permission: "<fill permission here>"
  - path: "/users"
    permission: "<fill permission here>"
  - path: "/admin"
    permission: "read:admin"
  - path: "/dashboard"
    permission: "read:dashboard"
    additionalFields:
      - ''
```

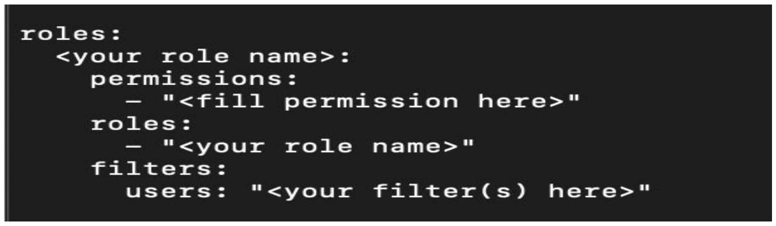
8. **Edit Configuration: `roles`:** In this section, you can define roles and permissions. These roles must match the roles defined in your DB. Roles can inherit permissions from other roles and can have filters applied to them.

   Use `read:` and `edit:` keywords for access levels:

   `read:<role>` grants view-only access.

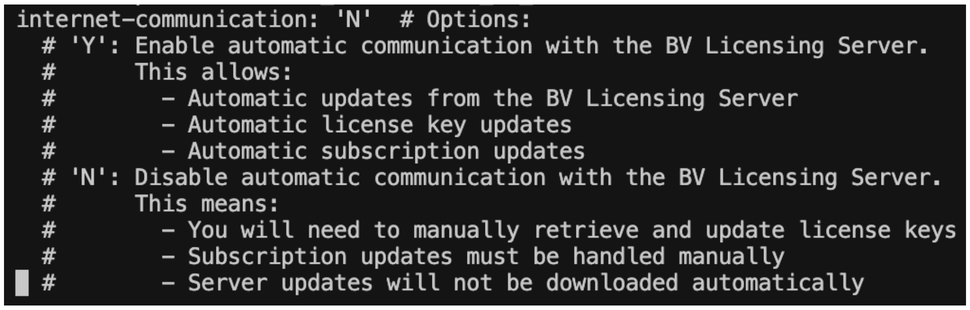
   `edit:<resource>` allows viewing and editing.

   You can also apply filters using valid MongoDB queries.



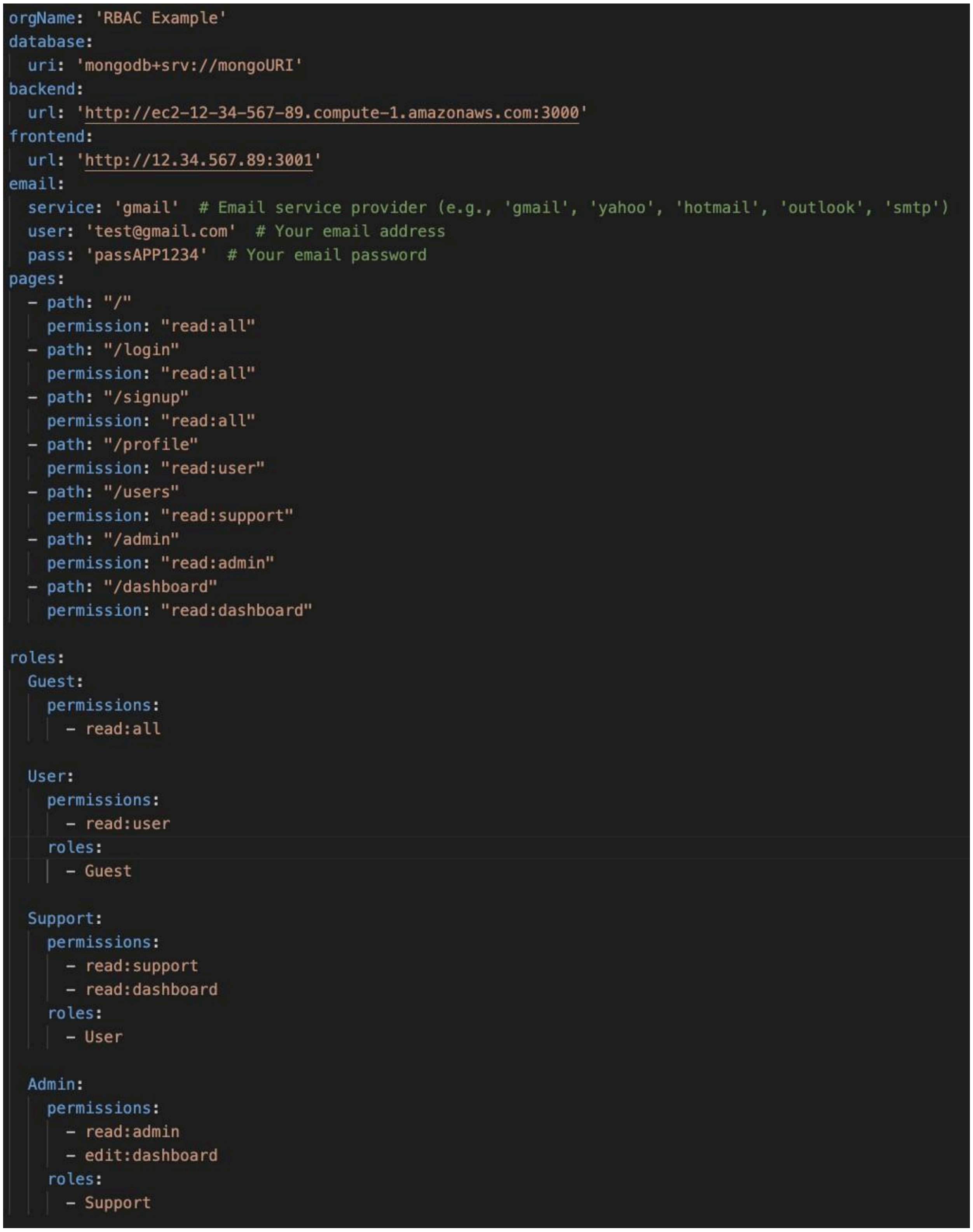
```
roles:
  <your role name>:
    permissions:
      - "<fill permission here>"
    roles:
      - "<your role name>"
    filters:
      users: "<your filter(s) here>"
```

9. **Edit Configuration: `internet-communication`:** In this section, you can choose whether you want the RBAC application to enable or disable internet communication to our BAIDEAC server. Default configuration is ‘N’.



```
internet-communication: 'N'  # Options:
  # 'Y': Enable automatic communication with the BV Licensing Server.
  #      This allows:
  #        - Automatic updates from the BV Licensing Server
  #        - Automatic license key updates
  #        - Automatic subscription updates
  # 'N': Disable automatic communication with the BV Licensing Server.
  #      This means:
  #        - You will need to manually retrieve and update license keys
  #        - Subscription updates must be handled manually
  #        - Server updates will not be downloaded automatically
```

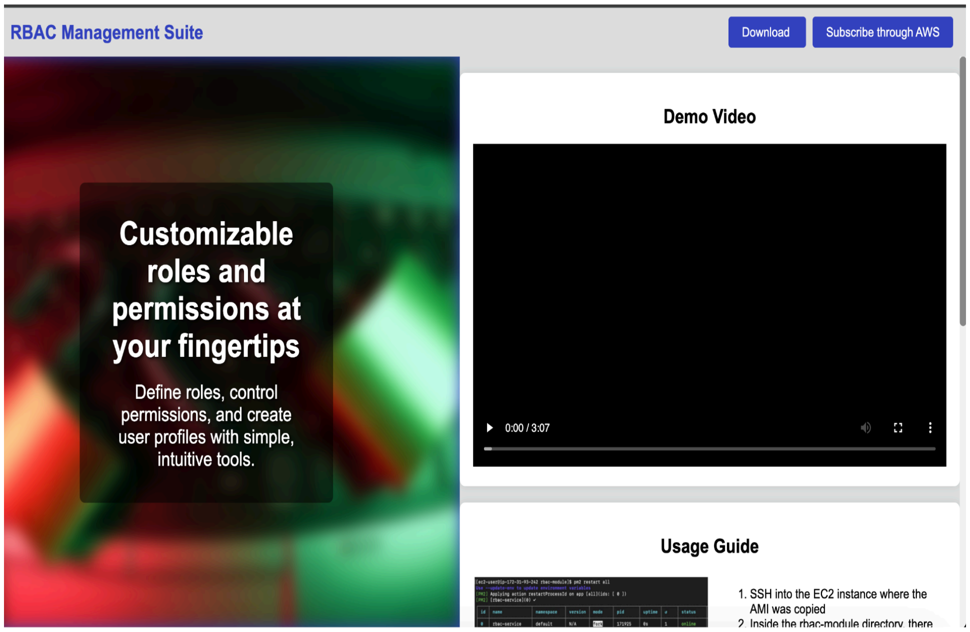
Example Configuration:



```yaml
orgName: 'RBAC Example'
database:
  uri: 'mongodb+srv://mongoURI'
backend:
  url: 'http://ec2-12-34-567-89.compute-1.amazonaws.com:3000'
frontend:
  url: 'http://12.34.567.89:3001'
email:
  service: 'gmail'  # Email service provider (e.g., 'gmail', 'yahoo', 'hotmail', 'outlook', 'smtp')
  user: 'test@gmail.com'  # Your email address
  pass: 'passAPP1234'  # Your email password
pages:
  - path: "/"
    permission: "read:all"
  - path: "/login"
    permission: "read:all"
  - path: "/signup"
    permission: "read:all"
  - path: "/profile"
    permission: "read:user"
  - path: "/users"
    permission: "read:support"
  - path: "/admin"
    permission: "read:admin"
  - path: "/dashboard"
    permission: "read:dashboard"

roles:
  Guest:
    permissions:
      - read:all

  User:
    permissions:
      - read:user
    roles:
      - Guest

  Support:
    permissions:
      - read:support
      - read:dashboard
    roles:
      - User

  Admin:
    permissions:
      - read:admin
      - edit:dashboard
    roles:
      - Support
```

Product Licensing

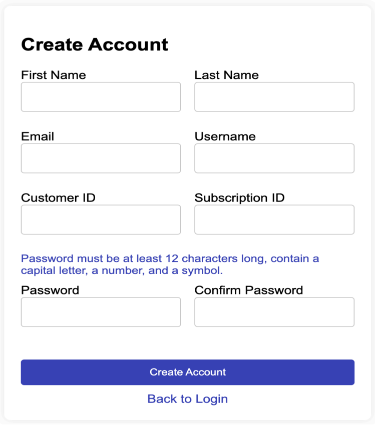
The RBAC product requires a license (paid or free subscription) to allow for the service to work. To obtain a license key, visit the product page and click the **Download** button. Here you can create an account for a paid license or free subscription version.



Customizable roles and permissions at your fingertips

Define roles, control permissions, and create user profiles with simple, intuitive tools.

1. SSH into the EC2 instance where the AMI was copied
2. Inside the rbac-module directory, there



Create Account

First Name	Last Name
Email	Username
Customer ID	Subscription ID
Password	Confirm Password

Password must be at least 12 characters long, contain a capital letter, a number, and a symbol.

Paid License: For first-time users that subscribed through the AWS Marketplace, follow these steps to create an account and obtain a license on the BAIDEAC Portal:

-> Go to the **Create Account** section and enter the required information:

- **Product ID**: Located on the AWS Agreement page for the agreement made when subscribing to the product.
- **Subscription ID**: Also found on the AWS Agreement page, this ID is identical to the Agreement ID.
- **Customer ID**: This is the AWS Customer ID associated with your account.

Using these details, you can create an account to retrieve the license key and complete the Licensing configuration. In your account, if your subscription is valid, you will see a table with your license key present.

Free Subscription License/Extensions:

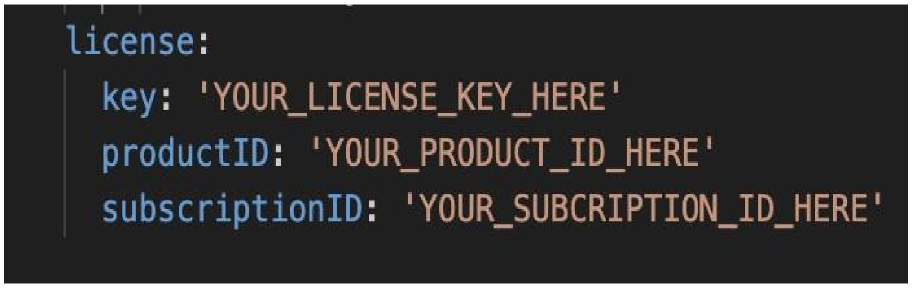
The user can also create a **Free Subscription Account** and **Request for Free Subscription License** as well. Using the same portal, you can create a free subscription account to retrieve the license key and complete the Licensing configuration. There is only 1 request allowed per user, and the *rbac-installer.tar* (from the external machine download instructions) is available for download under “Assets”.

If your free subscription license is expired, you will see an option to send a renewal request. This will send a request to the BAIDEAC support team for approval: upon approval, the license key will be updated in the portal and you will receive an email confirmation/denial.

RBAC Licensing Configuration

When first receiving the product, in the ‘**rbac.yaml**’ you will see the license section. Using the information below, please input these values in the yaml file.

a. **Product ID**: This is found in the AWS Agreement page for the agreement you made when subscribing to the product.
b. **Subscription ID**: This is found in the AWS Agreement page and is the same as the **Agreement ID**.



```
license:
  key: 'YOUR_LICENSE_KEY_HERE'
  productID: 'YOUR_PRODUCT_ID_HERE'
  subscriptionID: 'YOUR_SUBCRIPTION_ID_HERE'
```

After updating these values, proceed to the next step. The RBAC Module will validate the license key automatically if your subscription is active.

Restart the Server

Once the configuration is complete, restart the RBAC service to apply changes:

1. **Restart Command**:
   pm2 restart RBAC-EXE
2. **Check Logs**:
   pm2 log RBAC-EXE
3. **Verify Service**: Ensure that the service is running and the database is connected by checking the logs.

This completes the configuration of your RBAC Module. The following sections will detail how to integrate the service, RBAC features, and additional options to further customize your setup.

RBAC Commands:

Version Command: This command is used to report the version of the application. The container must be running for this command to work.

Inside the container: `/app/core/bin/rbac-service/rbac-module --version`

Outside the container: `docker exec -it rbac-container /app/core/bin/rbac-service/rbac-module --version`

Status Command: This command is used to report the status of the application. The container must be running for this command to work.

Inside the container: `/app/core/bin/rbac-service/rbac-module --status`

Outside the container: `docker exec -it rbac-container /app/core/bin/rbac-service/rbac-module --status`

RBAC Frontend Integration Guide

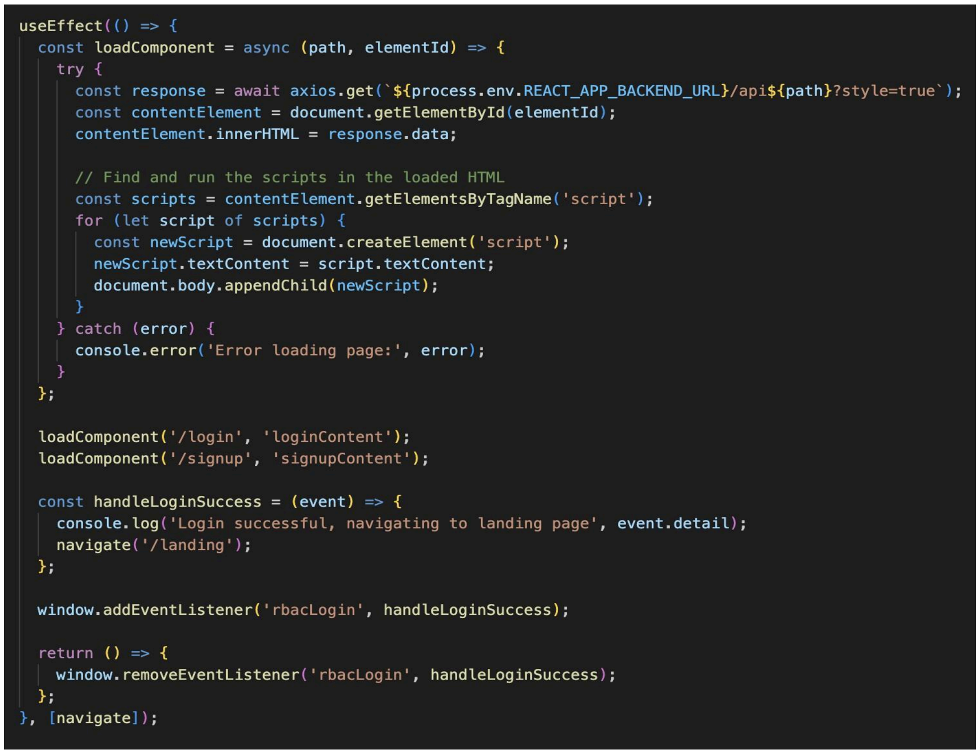
The RBAC Frontend Integration Guide provides step-by-step instructions for connecting your frontend application to the RBAC Module. This guide covers essential setup for login, signup, and accessing secure pages using the RBAC API and sample components.

Login/Signup

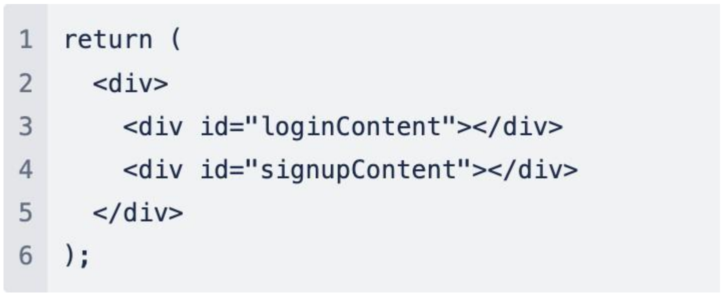
To enable login and signup functionalities with the RBAC Module, make an API call as follows:

http://<YOUR_IPV4>:3000/api${path}?style=true

For easy component loading, we recommend using a `**useEffect**` in the desired component. Using the `**loadComponent**` function, you can load any component from the RBAC Module by specifying the component's path and a unique `**elementID**` to reference it. You can repeat this process for as many components as needed, such as for the login and signup pages. Use the code below to reference how to implement:



```
useEffect(() => {
  const loadComponent = async (path, elementId) => {
    try {
      const response = await axios.get(`${process.env.REACT_APP_BACKEND_URL}/api${path}?style=true`);
      const contentElement = document.getElementById(elementId);
      contentElement.innerHTML = response.data;

      // Find and run the scripts in the loaded HTML
      const scripts = contentElement.getElementsByTagName('script');
      for (let script of scripts) {
        const newScript = document.createElement('script');
        newScript.textContent = script.textContent;
        document.body.appendChild(newScript);
      }
    } catch (error) {
      console.error('Error loading page:', error);
    }
  };

  loadComponent('/login', 'loginContent');
  loadComponent('/signup', 'signupContent');

  const handleLoginSuccess = (event) => {
    console.log('Login successful, navigating to landing page', event.detail);
    navigate('/landing');
  };

  window.addEventListener('rbacLogin', handleLoginSuccess);

  return () => {
    window.removeEventListener('rbacLogin', handleLoginSuccess);
  };
}, [navigate]);
```

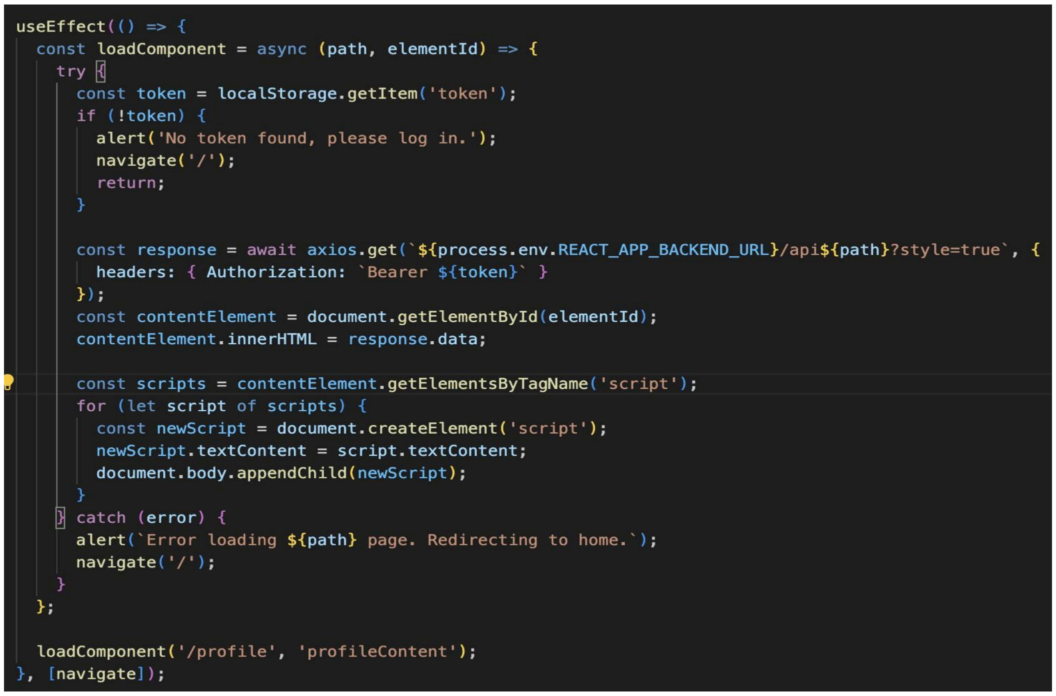


```
return (
  <div>
    <div id="loginContent"></div>
    <div id="signupContent"></div>
  </div>
);
```

During login, the user will receive an event listener that will notify them upon a successful login. This process will also generate a token stored in local storage, which will be required for accessing other RBAC-protected pages such as Profile and Dashboard.

Profile/Dashboard

For all pages other than Login/Signup, you must pass the token from local storage (generated during a successful login). Here is an example of how to set up this behavior in a `useEffect`:



```
useEffect(() => {
  const loadComponent = async (path, elementId) => {
    try {
      const token = localStorage.getItem('token');
      if (!token) {
        alert('No token found, please log in.');
        navigate('/');
        return;
      }

      const response = await axios.get(`${process.env.REACT_APP_BACKEND_URL}/api${path}?style=true`, {
        headers: { Authorization: `Bearer ${token}` }
      });
      const contentElement = document.getElementById(elementId);
      contentElement.innerHTML = response.data;

      const scripts = contentElement.getElementsByTagName('script');
      for (let script of scripts) {
        const newScript = document.createElement('script');
        newScript.textContent = script.textContent;
        document.body.appendChild(newScript);
      }
    } catch (error) {
      alert(`Error loading ${path} page. Redirecting to home.`);
      navigate('/');
    }
  };

  loadComponent('/profile', 'profileContent');
}, [navigate]);
```

Sample App

The RBAC Sample Application, demonstrated in the product video, is located in the root directory of your AMI copy, inside the **`rbac-client-repo`** directory. To connect the RBAC Module to the sample client app, follow these steps:

1. Navigate to the **`rbac-client-repo`** directory and open the **`.env`** file.
2. Update **`REACT_APP_BACKEND_URL`** to the IP + BACKEND PORT of your RBAC AMI instance (e.g., `http://<YOUR_IPV4>:3000`)
3. Run the following command to build the application:
   npm build
4. Navigate to the **`rbac-service`** directory and update the **`rbac.yaml`** file with the frontend URL. The URL should be set to:
   http://<YOUR_IPV4>:3001
5. Update the AWS Security Group to allow connections to the frontend port to access the site.
6. Restart all services:
   pm2 restart all
7. Visit the frontend URL in your browser.

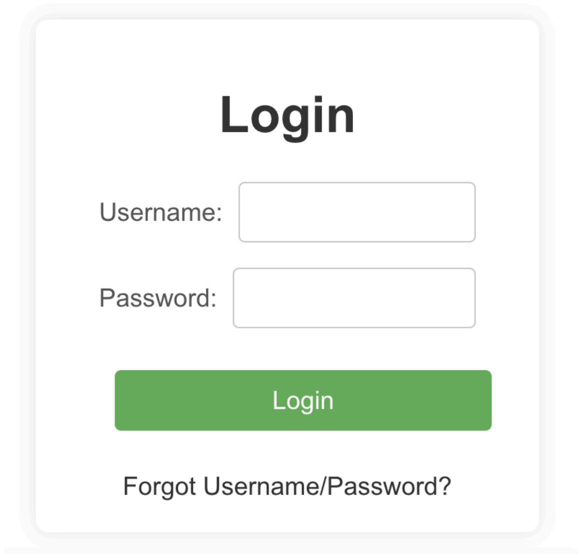
To stop the sample app, run the following command:
pm2 stop RBAC-CLIENT

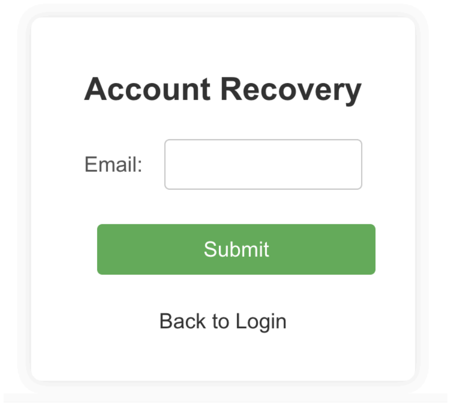
This guide should assist you in smoothly integrating the RBAC Module with your frontend and setting up the sample application.

RBAC Features

RBAC Component - Login

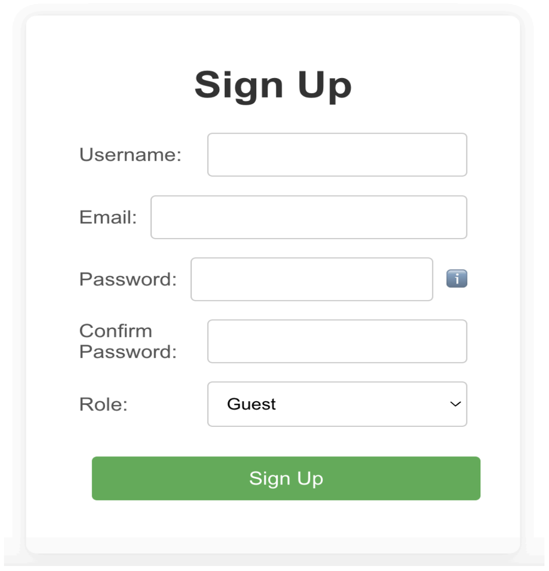
- Users can log in with unique usernames.
- Password validations are enforced, and a token is provided upon successful login.
- Includes "Forgot Username/Password" functionality, with an account recovery email sent to the user.





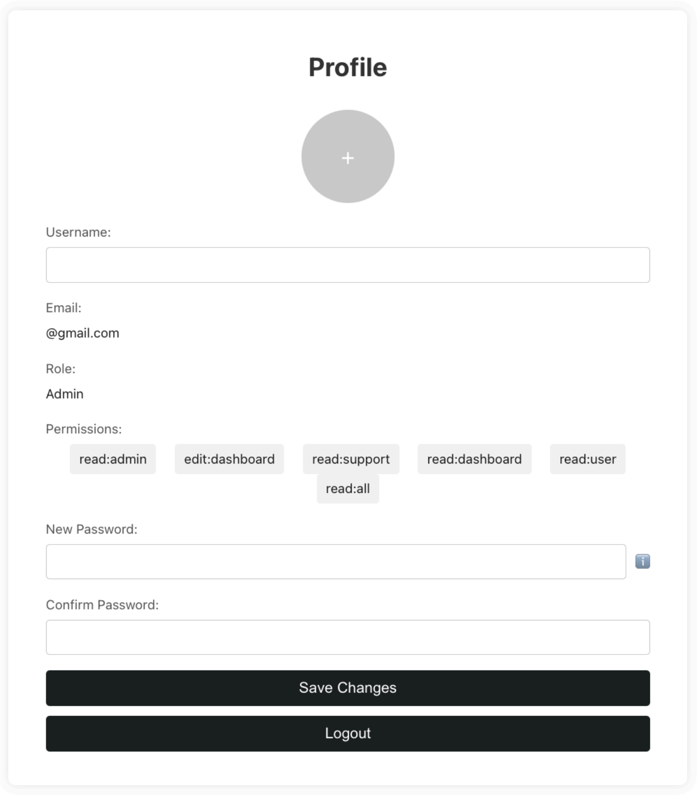
RBAC Component - Sign Up

- New users can sign up with roles configured in the `rbac.yaml` file.
- Password validations are displayed to guide users on secure password requirements.
- Email format verification ensures valid emails are entered during account creation.



RBAC Component - Profile

- ❖ Users can view essential information about their profile and assigned roles.
- ❖ Option to upload and save a profile picture.
- ❖ New password validations include strong password checks and verification against the last 10 passwords used.
- ❖ Logout button clears the login token, logging the user out securely.



Profile

Username:

Email:
@gmail.com

Role:
Admin

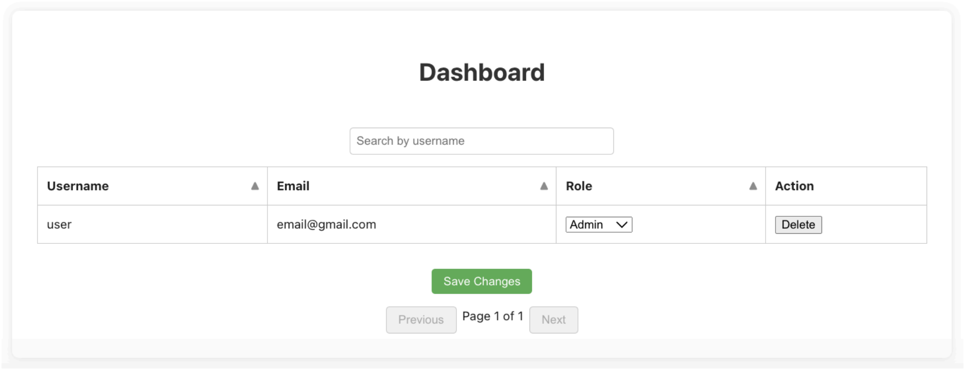
Permissions:
read:admin edit:dashboard read:support read:dashboard read:user read:all

New Password:

Confirm Password:

RBAC Component - Dashboard

- ❖ Allows viewing and searching of user information by username or email.
- ❖ Users with the appropriate role can edit other users' role types.
- ❖ Includes a delete user function with a confirmation pop-up to prevent accidental deletions.



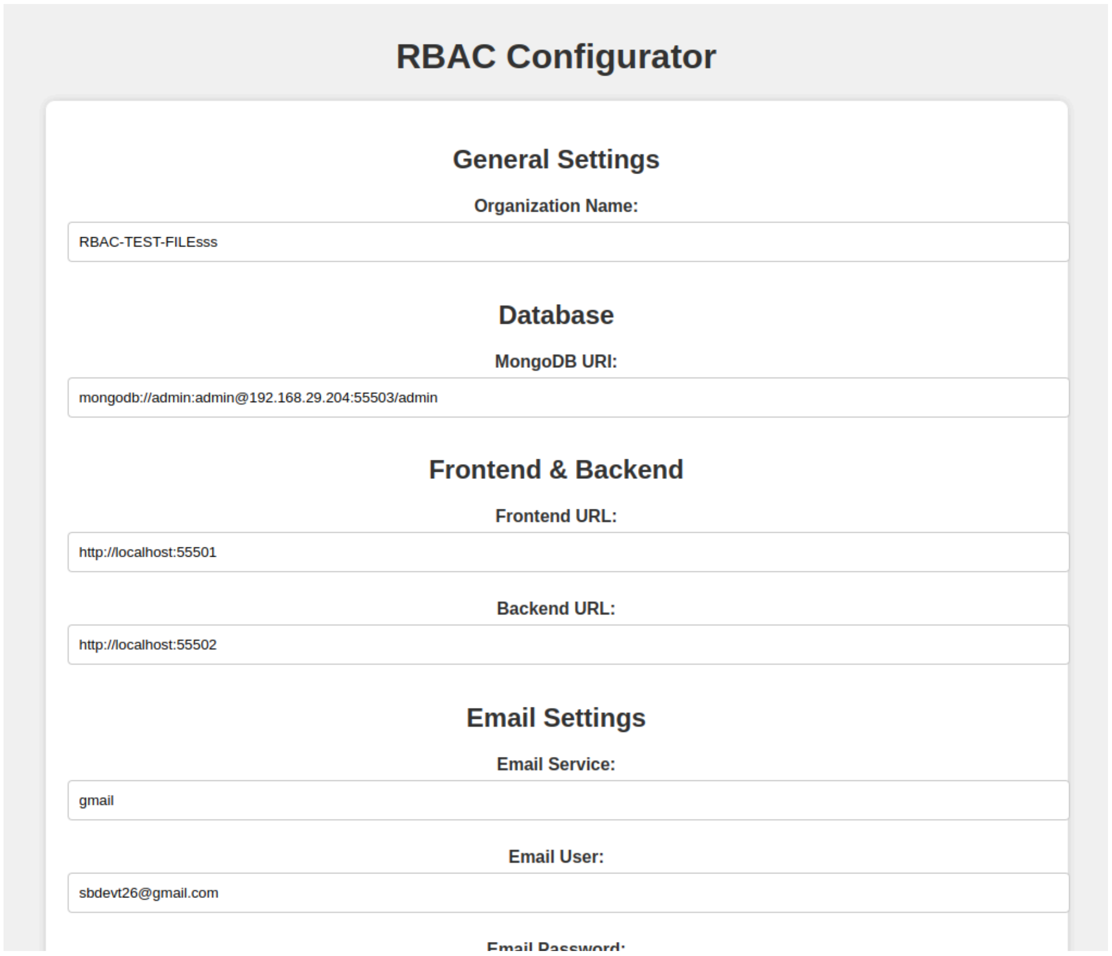
Dashboard

Search by username

Username	Email	Role	Action
user	email@gmail.com	Admin	Delete

RBAC Component – Setting

- ❖ Allows viewing and editing yaml file setting with new role setup



RBAC Configurator

General Settings

Organization Name: RBAC-TEST-FILEsss

Database

MongoDB URI: mongodb://admin:admin@192.168.29.204:55503/admin

Frontend & Backend

Frontend URL: http://localhost:55501

Backend URL: http://localhost:55502

Email Settings

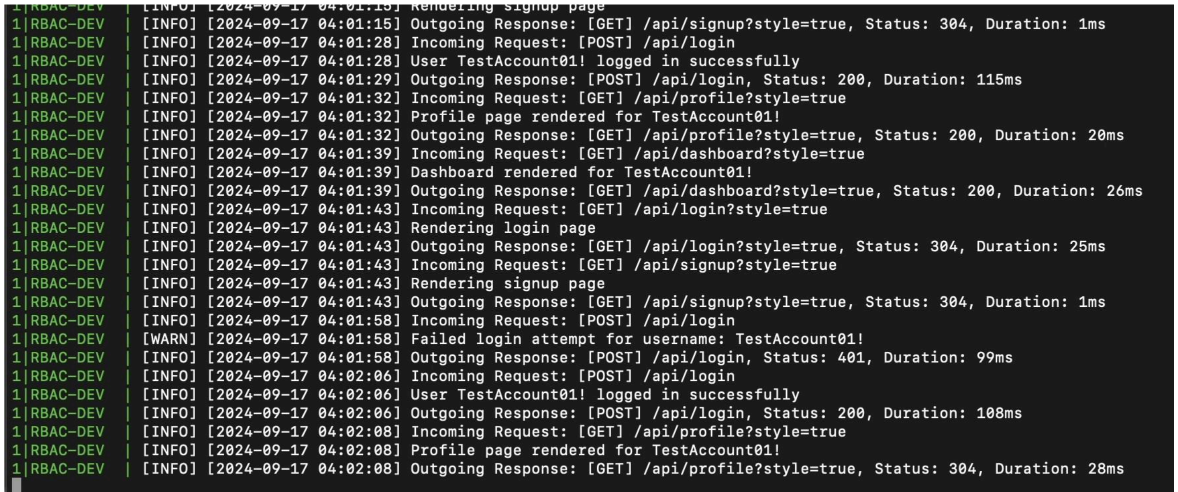
Email Service: gmail

Email User: sbdevt26@gmail.com

Email Password:

RBAC Module Logging

- ❖ View real-time logs on the AMI instance using `pm2 logs`.
- ❖ Logs are saved in a dedicated logs folder on the instance with automatic log rotation.
  - ➢ Logs are located at '**/core/logs**'
- ❖ Logs update in real time with each user action and are labeled with "Info," "Warn," and "Error" tags for efficient debugging.



```
1|RBAC-DEV  | [INFO] [2024-09-17 04:01:15] Rendering signup page
1|RBAC-DEV  | [INFO] [2024-09-17 04:01:15] Outgoing Response: [GET] /api/signup?style=true, Status: 304, Duration: 1ms
1|RBAC-DEV  | [INFO] [2024-09-17 04:01:28] Incoming Request: [POST] /api/login
1|RBAC-DEV  | [INFO] [2024-09-17 04:01:28] User TestAccount01! logged in successfully
1|RBAC-DEV  | [INFO] [2024-09-17 04:01:29] Outgoing Response: [POST] /api/login, Status: 200, Duration: 115ms
1|RBAC-DEV  | [INFO] [2024-09-17 04:01:32] Incoming Request: [GET] /api/profile?style=true
1|RBAC-DEV  | [INFO] [2024-09-17 04:01:32] Profile page rendered for TestAccount01!
1|RBAC-DEV  | [INFO] [2024-09-17 04:01:32] Outgoing Response: [GET] /api/profile?style=true, Status: 200, Duration: 20ms
1|RBAC-DEV  | [INFO] [2024-09-17 04:01:39] Incoming Request: [GET] /api/dashboard?style=true
1|RBAC-DEV  | [INFO] [2024-09-17 04:01:39] Dashboard rendered for TestAccount01!
1|RBAC-DEV  | [INFO] [2024-09-17 04:01:39] Outgoing Response: [GET] /api/dashboard?style=true, Status: 200, Duration: 26ms
1|RBAC-DEV  | [INFO] [2024-09-17 04:01:43] Incoming Request: [GET] /api/login?style=true
1|RBAC-DEV  | [INFO] [2024-09-17 04:01:43] Rendering login page
1|RBAC-DEV  | [INFO] [2024-09-17 04:01:43] Outgoing Response: [GET] /api/login?style=true, Status: 304, Duration: 25ms
1|RBAC-DEV  | [INFO] [2024-09-17 04:01:43] Incoming Request: [GET] /api/signup?style=true
1|RBAC-DEV  | [INFO] [2024-09-17 04:01:43] Rendering signup page
1|RBAC-DEV  | [INFO] [2024-09-17 04:01:43] Outgoing Response: [GET] /api/signup?style=true, Status: 304, Duration: 1ms
1|RBAC-DEV  | [INFO] [2024-09-17 04:01:58] Incoming Request: [POST] /api/login
1|RBAC-DEV  | [WARN] [2024-09-17 04:01:58] Failed login attempt for username: TestAccount01!
1|RBAC-DEV  | [INFO] [2024-09-17 04:01:58] Outgoing Response: [POST] /api/login, Status: 401, Duration: 99ms
1|RBAC-DEV  | [INFO] [2024-09-17 04:02:06] Incoming Request: [POST] /api/login
1|RBAC-DEV  | [INFO] [2024-09-17 04:02:06] User TestAccount01! logged in successfully
1|RBAC-DEV  | [INFO] [2024-09-17 04:02:06] Outgoing Response: [POST] /api/login, Status: 200, Duration: 108ms
1|RBAC-DEV  | [INFO] [2024-09-17 04:02:08] Incoming Request: [GET] /api/profile?style=true
1|RBAC-DEV  | [INFO] [2024-09-17 04:02:08] Profile page rendered for TestAccount01!
1|RBAC-DEV  | [INFO] [2024-09-17 04:02:08] Outgoing Response: [GET] /api/profile?style=true, Status: 304, Duration: 28ms
```

Troubleshooting/FAQ:

1. What if I want to change the ports in my configurations?
    a. Re-run the installation script and reassign the new ports. Your existing data will stay the same.
2. What if I want to use my own MongoDB Database?
    a. We configure a MongoDB for you in the installation process. This URI is provided in the Configuration Instructions above. If you wish to use your own MongoDB URI, simply add this to the RBAC.YAML file instead of the preconfigured MongoDB's URI. You can also stop the docker process named "rbac-mongo-db"
3. What if my License Key expires?
    a. If your license key is expired, please contact BAIDEAC Support for instructions on how to get the license key extended, if applicable.
4. How Secure is the BAIDEAC RBAC Management Suite?
    a. The application is tested for security vulnerabilities using Trivy for every release. BAIDEAC RBAC is proud to announce that from v1.5.0 release onwards there is no CVE reported in the scan report.