Roost on AWS

AWS configuration for Roost

Manual Deployment

Manual Deployment of Roost on AWS

1. High Level Architecture

Roost Ephemeral Environments as a Service (EaaS) platform provides a temporary, encapsulated deployment of a software application. Roost's Ephemeral environments provide robust, on-demand platforms for running tests, previewing features, and collaborating asynchronously across teams. Below is a high-level diagram of the AWS components required to deploy Roost on AWS.

1.1 Roost-AWS-Architecture.png

3. Prerequisites

Below are the infrastructure requirements for running Roost on AWS

Infrastructure Requirements

1. ALB with proper certificates
2. OAuth Details (Okta/ GoogleAuth etc.)
3. EC2 Instance (c5.2xlarge) x 3 and (t2.micro) x 1
4. RDS Database (AWS Aurora)
5. Execute Roost Control plane Script.

4. OAuth Provider Setup

Roost supports various authentication mechanisms as mentioned below

1. Github
2. Google
3. Microsoft
4. Linkedin
5. Okta

OKTA Auth Client Setup

- Sign in to your OKTA account with admin privileges (*If you do not have an existing Okta account, then sign-up at* Home | Okta Developer)
- From the left navigation menu, go to Applications -> Applications.
- Select Create App Integration → OIDC - OpenID Connect → Web Application, then click Next
- Fill in the suitable **App integration name,** upload the logo.
- Add **Sign-in redirect URIs**
  - https://<DNS_NAME>/login
- Allow Access to users thru Assignments → Controlled Access
  - Select the groups of users or Allow access to everyone
- Save and Make a note of the Okta Client ID and the Client Secret (It is needed later in the config below)
- From the left navigation menu, go to Security -> API
- Make a note of **Issuer URI** for default Authorisation Server
  - something like https://{your_domain}.okta.com/oauth2/default

Google Auth Client Setup

- Integrating Google Sign-In into your web app | Google Sign-In for Websites | Google Developers
- Login to https://console.cloud.google.com/apis/credentials
- Create Credentials, Select OAuth Client and Application Type as Web Application
- Add Authorised JavaScript Origin as

    - https://roostapi.roost.io:60001
    - https://<DNS_NAME
    - http://localhost:3000
    - http://localhost:4200
- Add Authorised redirect URIs
    - https://<DNS_NAME>/login
    - https://<DNS_NAME>/api/auth/redirect/google
    - https://roostapi.roost.io:60001/auth/redirect/google
- Download the JSON
- Make a note of the Google Client ID and the Client Secret (It is needed later in the config below)

5. Database Setup

Roost stores the status of the EaaS workflow and other relevant information in Database. Below are the steps to setup an Amazon Aurora DB in AWS

Amazon Aurora

1. Select RDS
2. Choose Create Database
3. Select “Easy Create” for “Amazon Aurora with MYSQL compatibility.”
4. Modify the RDS Security Group to allow TCP port 3306 access to the Control plane Instance security group only
5. Make a note of the writer instance database end-point, user, and password (It is needed later in the config below)
6. Create a new user with read-write privileges and avoid using an admin login.

```
# Sample command to create a user using MySQL CLI
# Provide password on prompt


mysql -h <SQL Host URL> -u <root|master|admin> -p
```

```
CREATE USER 'Roost'@'%' identified WITH mysql_native_password by 'Roost#123';
CREATE DATABASE roostio;
GRANT ALL on roostio.* to 'Roost'@'%';


# Execute the Roost Schema file, if available
\. /var/tmp/Roost/db/roost.sql
```