

Roost on AWS

AWS configuration for Roost

- Manual Deployment
 - 1. High Level Architecture
 - 3. Prerequisites
 - 4. OAuth Provider Setup
 - 5. Database Setup
- Using Terraform
 - 1. Getting Started
 - 2. OAuth Provider Setup
 - 3. Terraform variables

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
```

Using Terraform

Deploy Roost on AWS using Terraform

1. Getting Started

Roost provides terraform scripts to spin up and configure the EaaS platform easily. Below are the steps for deploying Roost on AWS using Terraform

High-Level Architecture

Roost Terraform scripts create the below components in the AWS cloud

Roost-AWS-Architecture.png

Prerequisites

- User Account with Admin privileges / Policies enabled to run terraform scripts
- region
- route53_hosted_zone_id
- ec2_ami
- enterprise_dns
- ip_block_vpc (VPC CIDR where Roost would be set up)
- okta_client_id or appropriate auth provider (please refer [link](#) here)
- company

Clone the Repo

```
https://github.com/roost-io/terraform.git
```

2. 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)

3. Terraform variables

Please follow the below steps to modify terraform files to incorporate the

- Clone the GitHub repo.

```
git clone https://github.com/roost-io/terraform.git
```

- Copy terraform.tfvars.original as terraform.tfvars
- Fill in the below details

```
region
route53_hosted_zone_id
ec2_ami
enterprise_dns
ip_block_vpc (VPC CIDR where Roost would be setup)
okta_client_id or appropriate auth provider
company (Provided by Roost team)
```

Terraform Variable Definitions

Field	Values	Description
prefix	"terraform-eaas"	
region	"us-west-1"	
az1_suffix	"b"	
az2_suffix	"c"	
deletion_protection	false	
route53_hosted_zone_id		
enterprise_dns	"eaas.example.com"	
ec2_ami	"ami-03df6dea56f8aa618"	

key_pair	"roost-eaas-keypair"	
generate_key_pair	true	
device_name	"sdh"	
ip_block_vpc	"172.32.255.192"	
instance_type_controlplane	"t3.large"	
instance_type_jumphost	"t3.micro"	
google_client_id		
google_client_secret		
github_client_id		
github_client_secret		
linkedin_client_id		
linkedin_client_secret		
azure_client_id		
azure_client_secret		
okta_client_id	"00a4bweaxcqn2sfTu5d7"	
okta_client_secret	"D5oRtWXUWcl9gp1312dVtuSoumU4vrECO4wSsqAO"	
okta_issuer		
roost_jwt_token		
company		
company_logo	"https://roost.ai/hubfs/logos/Roost.ai-logo-gold.svg"	
enterprise_email_domain	"example.com"	

admin_email	"admin@email"	
senders_email	"sender@email"	
is_own_mysql	false	
mysql_host	"mysql_db_host_url"	
mysql_password	"Roost#123"	
mysql_username	"Roost"	
mysql_port	3306	
mysql_db_name	"roostio"	
mysql_root_password	"Admin#123"	
senders_email_pass		
email_smtp_host		