

# Deployment Guide

This guide documents how to deploy the system onto AWS servers. There are two methods of deployment: Using the Terraform script or manually deploying. The Terraform script is intended to automate the process of deployment so it can be integrated with Github Actions, though if something is broken it may be more convenient to use the manual deployment procedure, so both are included in this guide.

<b>Deployment Guide.....</b>	<b>1</b>
Manual Deployment Guide.....	2
Provision Infrastructure.....	2
ECR (Elastic Container Registry):.....	2
ECS (Elastic Container Service).....	2
RDS (Relational Database Service).....	3
Cognito.....	3
S3 (Simple Storage Service).....	4
SNS (Simple Notification Service).....	4
Terraforming Guide.....	6
Stage 1 - Setup AWS Terraform User.....	6
Stage 2 - Terraform Setup.....	13
Google Auth Setup.....	14
Terraform Continued.....	19
Push App Image.....	19
Extra Steps.....	19
Integrate with Github Actions.....	20
Useful Commands.....	20

# Manual Deployment Guide

Note: Currently we have Terraform scripts to automatically provision all the architecture, however, for this guide we will include manual provisioning instructions.

The project is currently deployed at <http://www.roommatebudgethelper.tk:3000/>

## Provision Infrastructure

In order for the application to work correctly, all infrastructure must be provisioned on AWS. The project will be using the following AWS Services:

- ECR: Elastic container repository for hosting the docker app image
- ECS: Elastic container service to run docker application
- RDS: relational database service for hosting postgres database
- Cognito User Pool: Used for managing registration and login and storing user information
- S3: Simple storage service for hosting file uploads such as images
- SNS: Simple notification service used for sending email and push notifications

Create a root account on AWS. You should create IAM users for each developer / maintainer of the project that only allows the AWS permissions they need to modify any infrastructure. Within your IAM user account complete the following steps to provision all infrastructure needed for the project:

## ECR (Elastic Container Registry):

1. From the AWS console, search for the Elastic container registry
2. Click Create repository
3. Chose private visibility and give it a unique name such as "roommate-budget-helper-nextjs-application"
4. Enable KMS encryption and click create repository

Uploading the image to ecr:

1. From aws console, select the repository you just created
2. Click on "view push commands"
3. Follow instructions to build and push the container (note you must be in the app folder on the project to run the docker build command)

## ECS (Elastic Container Service)

1. Navigate to AWS console and search for Elastic Container service
2. Click on Task Definitions
3. Select create new task definition with JSON
  - a. Within the terraform folder on RBH-25-terraform-and-cd-setup navigate to the task-definition folder

- b. Copy paste the service template
- c. Replace the following fields
  - i. Network mode = awsvpc
  - ii. Port = 3000
  - iii. Container\_name = name of your image in ECR
  - iv. Memory = 512mb or the size of your target EC2 instance
  - v. Launch type = EC2
  - vi. Log\_group = roommate-budget-helper-prod
  - vii. Env = prod
  - viii. Region = us-east-1

## RDS (Relational Database Service)

1. Navigate to AWS console and search for RDS
2. Click create database
3. Select standard create
4. Select PostgreSQL as engine type
5. Use production template
6. Select a deployment option that matches your intended cost
7. Name the DB cluster roommate-budget-helper-prod
8. Create a username for the database
9. Select auto generate a password
10. Select a size of DB within your price range such as db.t3.micro
11. Click create database

## Cognito

1. Navigate to AWS console and search for Cognito
2. Click create user pool
3. Select the following sign up options
  - a. User name
  - b. Email
4. Click next
5. Use Cognito defaults for password policy
6. Select Optional MFA for multi-factor authentication
  - a. Allow for authenticator apps
7. Use default user account recover settings and click next
8. Scroll to bottom and click next again
9. Select Send email with Cognito and press next
10. Name the userpool rbh-prod
11. Select public client and give it the name RBH-client
12. Don't generate a client secret and press next
13. Select create user pool

## S3 (Simple Storage Service)

1. Navigate to AWS console and search for S3
2. Click create bucket
3. For the bucket name enter something globally unique such as roommate-budget-helper-production-s3
4. Select US East 1 for the region
5. Select ACLs disabled
6. Make sure block all public access is selected
7. Enable encryption
8. Press create bucket
9. From the buckets page select the newly created bucket
10. Click on permissions
11. Under Cross-origin resource sharing (CORS) select edit
12. Paste the following json:

```
[
  {
    "AllowedHeaders": [
      "*"
    ],
    "AllowedMethods": [
      "PUT",
      "POST",
      "DELETE"
    ],
    "AllowedOrigins": [
      "*"
    ],
    "ExposeHeaders": []
  }
]
```

13. Replace allowed origins with the intended domain of the production application
14. Click save changes

## SNS (Simple Notification Service)

1. Navigate to AWS console and search for SNS
2. Navigate to topics
3. Click create topic
4. Select FIFO
5. Enter the name invite-roommate
6. Enter the display name Roommate Invitations
7. Turn on message deduplication

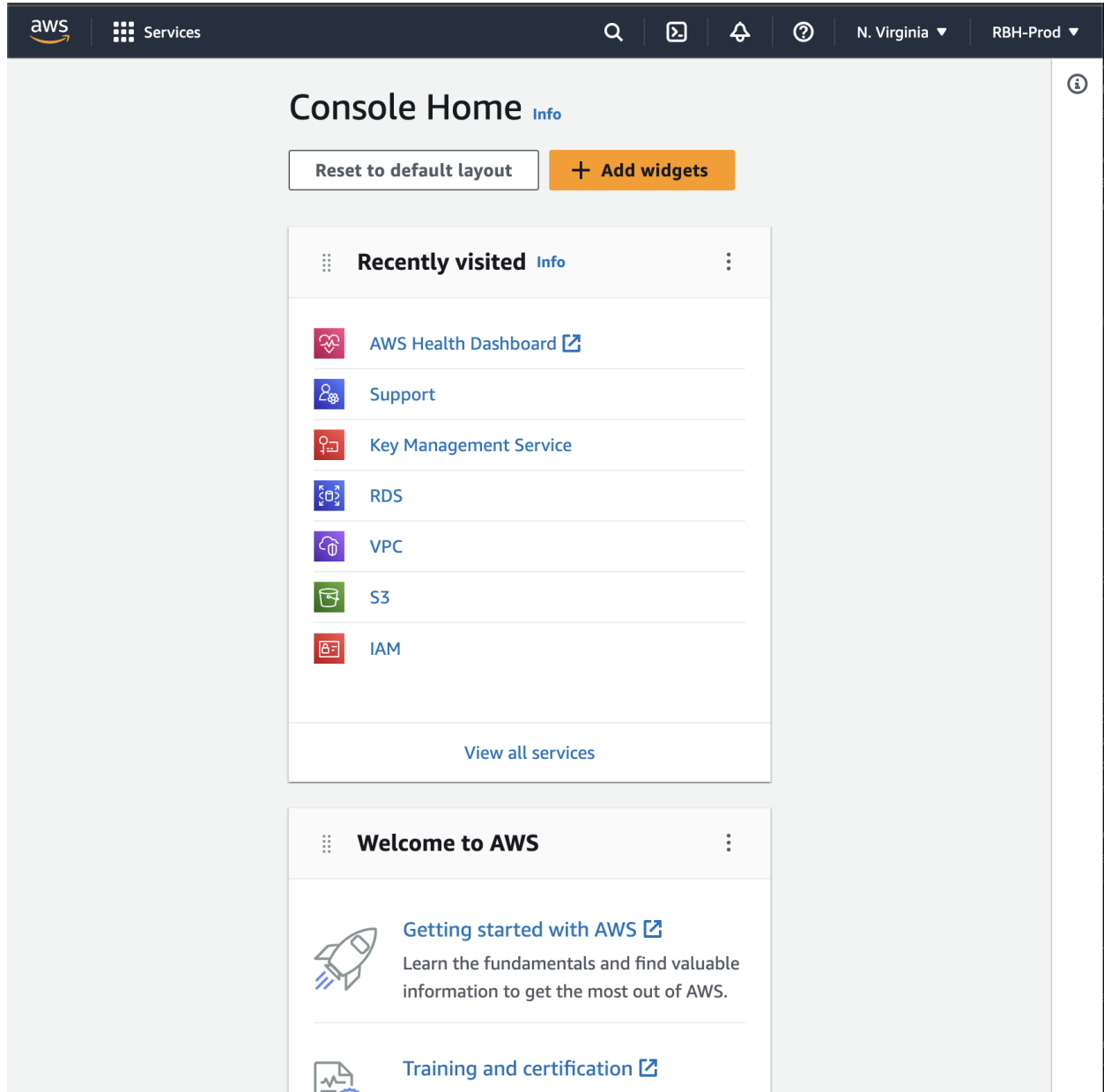
8. Enable Encryption with default CMK
9. Select create topic
10. Repeat steps 2-9 with the name reminders and display name Reminder subsystem

All required infrastructure should now be provisioned and deployed.

# Terraforming Guide

## Stage 1 - Setup AWS Terraform User

### 1. Login to AWS



2. Ensure us-east-1 is selected for the region

The screenshot shows the AWS Management Console Home page. The top navigation bar includes the AWS logo, a 'Services' menu, and a search bar. The current region is 'N. Virginia' and the account is 'RBH-Prod'. The main content area is titled 'Console Home' and features a 'Reset to default layout' button. Below this is a 'Recently visited' section with links to AWS Health Dashboard, Support, Key Management Service, RDS, VPC, S3, and IAM. A 'Welcome to AWS' section includes links for 'Getting started with AWS' and 'Training and certification'. On the right side, a dropdown menu for regions is open, showing a list of regions and their IDs. The 'us-east-1' region is highlighted. A note at the bottom of the dropdown states: 'There are 10 Regions that are not enabled for this account'.

Region	Region ID
US East (N. Virginia)	us-east-1
US East (Ohio)	us-east-2
US West (N. California)	us-west-1
US West (Oregon)	us-west-2
Asia Pacific (Mumbai)	ap-south-1
Asia Pacific (Osaka)	ap-northeast-3
Asia Pacific (Seoul)	ap-northeast-2
Asia Pacific (Singapore)	ap-southeast-1
Asia Pacific (Sydney)	ap-southeast-2
Asia Pacific (Tokyo)	ap-northeast-1
Canada (Central)	ca-central-1
Europe (Frankfurt)	eu-central-1
Europe (Ireland)	eu-west-1
Europe (London)	eu-west-2
Europe (Paris)	eu-west-3
Europe (Stockholm)	eu-north-1
South America (São Paulo)	sa-east-1

There are 10 Regions that are not enabled for this account

### 3. Navigate to IAM and select the IAM service

The screenshot shows the AWS Management Console interface. At the top, the AWS logo and 'Services' menu are visible. A search bar contains the text 'iam'. Below the search bar, a horizontal navigation bar lists categories: Services (9), Features (19), Resources (with a 'New' badge), Blogs (1,551), Documentation (45,691), and Knowledge Articles (16). The main content area displays a list of search results for 'iam'. The first result, 'IAM', is highlighted with a blue border. It includes an icon, the name 'IAM' with a star, and the description 'Manage access to AWS resources'. Other results listed below include 'IAM Identity Center (successor to AWS Single Sign-On)', 'Resource Access Manager', 'Serverless Application Repository', 'Application Composer', 'AWS Proton', 'Amazon MQ', 'Elastic Container Service', and 'Service Catalog', each with its respective icon, name, star, and description.

aws Services 🔍 📄 🔔 ⓘ N. Virginia ▼ RBH-Prod ▼

iam ✕ Cancel

Services (9) Features (19) Resources **New** Blogs (1,551) Documentation (45,691) Knowledge Articles (16)

- IAM** ☆  
Manage access to AWS resources
- IAM Identity Center (successor to AWS Single Sign-On)** ☆  
Manage workforce user access to multiple AWS accounts and cloud applications
- Resource Access Manager** ☆  
Share AWS resources with other accounts or AWS Organizations
- Serverless Application Repository** ☆  
Assemble, deploy, and share serverless applications within teams or publicly
- Application Composer** ☆  
Visually design and build serverless applications quickly
- AWS Proton** ☆  
Manage your infrastructure so developers can focus on coding.
- Amazon MQ** ☆  
Managed message broker service for Apache ActiveMQ and RabbitMQ
- Elastic Container Service** ☆  
Highly secure, reliable, and scalable way to run containers
- Service Catalog** ☆  
Create, share, organize, and govern your curated infrastructure as code (IaC) templates



4. Select “Users” from the menu on the left of the IAM dashboard

The screenshot shows the AWS IAM dashboard interface. On the left, the navigation menu is expanded, and the 'Users' link is highlighted with a red arrow. The main content area displays the 'IAM dashboard' title, followed by 'Security recommendations' with a red notification badge showing '1'. Below this, there are two security alerts: 'Add MFA for root user' (with a red warning icon) and 'Root user has no active access keys' (with a green checkmark icon). Further down, the 'IAM resources' section contains a table with four columns: 'User groups' (0), 'Users' (1), 'Roles' (2), and 'Policies' (0). At the bottom of the main content area, there is a 'What's new' section with a list of updates for features in IAM, including 'Advanced Notice: Amazon S3 will automatically enable S3 Block Public Access for all new buckets starting in April 2023', 'AWS IAM Identity Center now supports session management capabilities for CLI and SDKs', 'AWS Lambda announces support for Attribute-Based Access Control (ABAC)', and 'Amazon ElastiCache simplifies password rotations with Secrets Manager'. A 'more' link is provided at the end of the list. The left-hand navigation menu includes sections for 'Access management' (User groups, Users, Roles, Policies, Identity providers, Account settings) and 'Access reports' (Access analyzer, Archive rules, Analyzers, Settings, Credential report, Organization activity, Service control policies (SCPs)). At the bottom of the navigation menu, there are links for 'Related consoles' including 'IAM Identity Center' (marked as 'New') and 'AWS Organizations'.

**Identity and Access Management (IAM)**

Search IAM

**Dashboard**

▼ **Access management**

- User groups
- Users**
- Roles
- Policies
- Identity providers
- Account settings

▼ **Access reports**

- Access analyzer
  - Archive rules
  - Analyzers
  - Settings
- Credential report
- Organization activity
- Service control policies (SCPs)

*Related consoles*

- [IAM Identity Center](#) **New**
- [AWS Organizations](#)

## IAM dashboard

### Security recommendations 1

**⚠ Add MFA for root user**  
Add MFA for root user - Enable multi-factor authentication (MFA) for the root user to improve security for this account.

**✅ Root user has no active access keys**  
Using access keys attached to an IAM user instead of the root user improves security.

### IAM resources

User groups	Users	Roles	Policies
0	1	2	0

### What's new [↗](#)

Updates for features in IAM

- [Advanced Notice: Amazon S3 will automatically enable S3 Block Public Access for all new buckets starting in April 2023.](#) 5 months ago
- [AWS IAM Identity Center now supports session management capabilities for CLI and SDKs.](#) 6 months ago
- [AWS Lambda announces support for Attribute-Based Access Control \(ABAC\)](#) 6 months ago
- [Amazon ElastiCache simplifies password rotations with Secrets Manager.](#) 6 months ago

[more](#)

5. Press “Add user” button

The screenshot shows the AWS IAM console interface. At the top, there's a navigation bar with the AWS logo, 'Services', and a search bar. Below the navigation bar, a green notification banner states 'User Terraform deleted.' with a close button. The left sidebar is titled 'Identity and Access Management (IAM)' and contains a search bar and a list of navigation items: 'Dashboard', 'Access management' (expanded), 'Users' (selected), 'Roles', 'Policies', 'Identity providers', 'Account settings', 'Access reports' (expanded), 'Access analyzer', 'Archive rules', 'Analyzers', 'Settings', 'Credential report', 'Organization activity', and 'Service control policies (SCPs)'. At the bottom of the sidebar, there are 'Related consoles' links for 'IAM Identity Center' (marked 'New') and 'AWS Organizations'. The main content area is titled 'IAM > Users'. It features a 'Users (0)' header with an 'Info' link and a description: 'An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.' Below this is a search bar with the placeholder 'Find users by username or access key'. To the right of the search bar are buttons for 'Refresh', 'Delete', and 'Add users'. A red arrow points to the 'Add users' button. Below the search bar is a pagination control showing '< 1 >' and a settings gear icon. At the bottom, there's a table header with columns for 'User name' and 'Groups'.

6. Make the username "Terraform" and click next

aws Services 🔍 📄 🔔 ⓘ Global ▼ RBH-Prod ▼

☰ IAM > Users > Create user ⓘ

Step 1  
**Specify user details**

Step 2  
Set permissions

Step 3  
Review and create

## Specify user details

### User details

User name

Terraform

The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and + = , . @ \_ - (hyphen)

☐ Provide user access to the AWS Management Console - *optional*

If you're providing console access to a person, it's a [best practice](#) to manage their access in IAM Identity Center.

ⓘ If you are creating programmatic access through access keys or service-specific credentials for AWS CodeCommit or Amazon Keyspaces, you can generate them after you create this IAM user. [Learn more](#)

Cancel Next

7. Select “Attach policies directly” and select the “AdministratorAccess” AWS managed policy, then press next

The screenshot shows the 'Set permissions' step in the AWS IAM console. Under 'Permissions options', 'Attach policies directly' is selected. Below, a table lists 4 matching policies, with 'AdministratorAccess' selected. The 'Next' button is highlighted with a red arrow.

Policy name	Type	Attached entities
<input checked="" type="checkbox"/> AdministratorAccess	AWS managed - job function	1
<input type="checkbox"/> AdministratorAccess-Amplify	AWS managed	0
<input type="checkbox"/> AdministratorAccess-AWSElasticBeanstalk	AWS managed	0
<input type="checkbox"/> AWSAuditManagerAdministratorAccess	AWS managed	0

8. Confirm the user creation on the next screen
9. Click on the newly created key user and navigate to “Security Credentials”, scroll down and click on “Create access key”

The screenshot shows the 'Security Credentials' tab for the user 'Terraform'. It displays summary information, console sign-in status, MFA status, and access keys. The 'Create access key' button is highlighted with a red arrow.

Device type	Identifier	Created on
No MFA devices. Assign an MFA device to improve the security of your AWS environment.		

10. Select “Command Line Interface (CLI)” and accept the above recommendation, then press “next”

The screenshot shows the AWS IAM console 'Create access key' page. The breadcrumb trail is IAM > Users > Terraform > Create access key. The page is divided into three steps: Step 1 (Access key best practices & alternatives), Step 2 (optional, Set description tag), and Step 3 (Retrieve access keys). The main content area is titled 'Access key best practices & alternatives' and contains a warning about long-term credentials. Below this, there are six radio button options: 'Command Line Interface (CLI)', 'Local code', 'Application running on an AWS compute service', 'Third-party service', 'Application running outside AWS', and 'Other'. The 'Command Line Interface (CLI)' option is selected. Below these options is a section titled 'Alternatives recommended' with two bullet points: 'Use AWS CloudShell, a browser-based CLI, to run commands.' and 'Use the AWS CLI V2 and enable authentication through a user in IAM Identity Center.' At the bottom, there is a checkbox labeled 'I understand the above recommendation and want to proceed to create an access key.' which is checked. A 'Next' button is located at the bottom right of the page.

11. Press “Create access key”

12. On the next screen, it will provide you the credentials. Store them safely and press “Done”

Your AWS account should now be configured correctly for terraform

## Stage 2 - Terraform Setup

1. On the cloned repository open a terminal
2. navigate to the “terraform” directory
3. Run “terraform init”

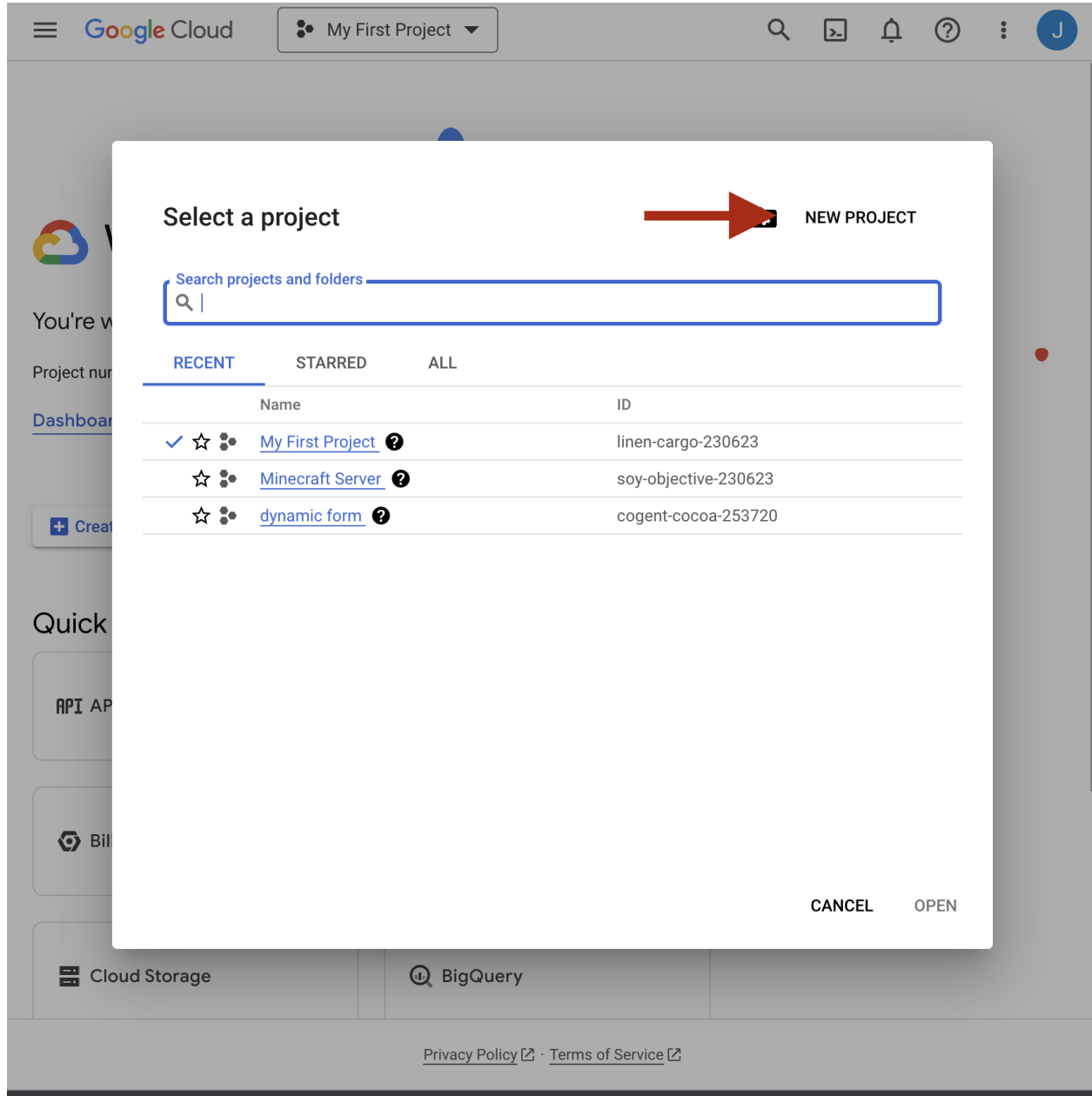
4. Run “aws configure”
5. Enter the access key of your Terraform user created in stage 1
6. Enter the secret key for the terraform user
7. Setup the default region as “us-east-1”
8. Leave the output format as default

## Google Auth Setup

Before continuing with Terraform, you must create google oauth credentials

1. Navigate to “<https://console.cloud.google.com/>”
2. Login with the Google account you plan on using to manage the project

### 3. Create a new project



#### 4. Enter a project name and press next

### New Project



You have 11 projects remaining in your quota. Request an increase or delete projects. [Learn more](#)

[MANAGE QUOTAS](#)

Project name \*

RBH-production



Project ID: rbh-production. It cannot be changed later. [EDIT](#)

Location \*

No organization

[BROWSE](#)

Parent organization or folder

[CREATE](#)

[CANCEL](#)



5. Once the project is created and is the actively selected project. Navigate to “APIs & Services”

Google Cloud RBH-production

APIs & Services + ENABLE APIS AND SERVICES

1 hour 6 hours 12 hours ✓ 1 day 2 days 4 days 7 days 14 days 30 days

Traffic

No data is available for the selected time frame.

6:00 AM 12:00 PM

UTC-4 May 10 6:00 AM 12:00 PM

Median latency

Enabled APIs & services  
Library  
Credentials  
OAuth consent screen  
Page usage agreements

https://console.cloud.google.com/apis?project=rbh-production

6. From the left side menu select credentials

Google Cloud RBH-production

APIs & Services

Enabled APIs & services

Library

**Credentials**

OAuth consent screen

Page usage agreements

Credentials + CREATE CREDENTIALS DELETE

Create credentials to access your enabled APIs. [Learn more](#)

Remember to configure the OAuth consent screen with information about your application.

CONFIGURE CONSENT SCREEN

API Keys

<input type="checkbox"/>	Name	Creation date ↓	Restrictions	Actions
No API keys to display				

OAuth 2.0 Client IDs

<input type="checkbox"/>	Name	Creation date ↓	Type	Client ID	Actions
No OAuth clients to display					

Service Accounts [Manage service accounts](#)

<input type="checkbox"/>	Email	Name ↑	Actions
No service accounts to display			

7. Click "Configure Consent Screen"
8. Select "External" and press create
9. Fill in the information with App name, user email, app logo, and domain
10. Click continue
11. Under scopes, select userinfo.email, userinfo.profile, and openid
12. Press continue and continue
13. Press return to dashboard
14. Return to credentials
15. Click create credentials
16. Select "OAuth Client ID"

17. Select "Web Application"
18. Name it "RBH Web"
19. Add the following javascript origins
  - a. "https://<your-domain-here>/api/auth/callback/google"
  - b. "http://localhost:3000/api/auth/callback/google" for local development
20. Press create, it will provide you the client id and client secret, store them somewhere safe and feel free to download the JSON file.

## Terraform Continued

9. In the terraform folder in your terminal, run "terraform plan"
10. Enter the google id and secret
11. Verify "terraform creates" a valid plan
12. run terraform apply to provision the infrastructure
13. Type "yes" to approve the creation
14. Log back into AWS after the infrastructure is created
15. Navigate to EC2
16. Create a new elastic IP
17. Attach it to the running ec2 instance

## Push App Image

1. Navigate to app folder of the repository in your terminal
2. Log into AWS and navigate to ECR
3. Click on "rbh-app" under private repositories
4. Click view push commands
5. Run each of the push commands

## Extra Steps

1. Configure your domain to point to a nginx reverse proxy (host yourself)
2. Setup SSL on Nginx reverse proxy
3. Reverse proxy should redirect http, https of your domain to the elastic ip of your instance in AWS targeting port 3000

## Integrate with Github Actions

1. Set the following Secrets in Github Actions on the repository
  - a. AWS\_ACCESS\_KEY\_ID = <access key of terraform user>
  - b. AWS\_SECRET\_ACCESS\_KEY = <secret key of terraform user>
  - c. COGNITO\_CLIENT\_ID = <from terraform output>
  - d. COGNITO\_USER\_POOL = <from terraform output>
  - e. CYPRESS\_API\_KEY = <from mailslurp - Read Dev Guide 'Setting up testing'>
  - f. DATABASE\_URL = <from terraform output>
  - g. GOOGLE\_CLIENT\_ID = <from google setup>
  - h. GOOGLE\_CLIENT\_SECRET = <from google setup>yum install certbot  
python3-certbot-nginx
  - i. NEXTAUTH\_SECRET = <from terraform output>
  - j. NEXTAUTH\_URL = <base url of domain e.x https://roommatebudgethelper.tk>
  - k. S3\_BUCKET\_NAME = <from terraform output>
  - l. SLACK\_TOKEN = <see make slack token>
  - m. URL = <base url of domain e.x https://roommatebudgethelper.tk>

## Useful Commands

1. "terraform output <name of output>"
  - a. Provides the value of terraform output field, will show sensitive information