

Steps to follow.

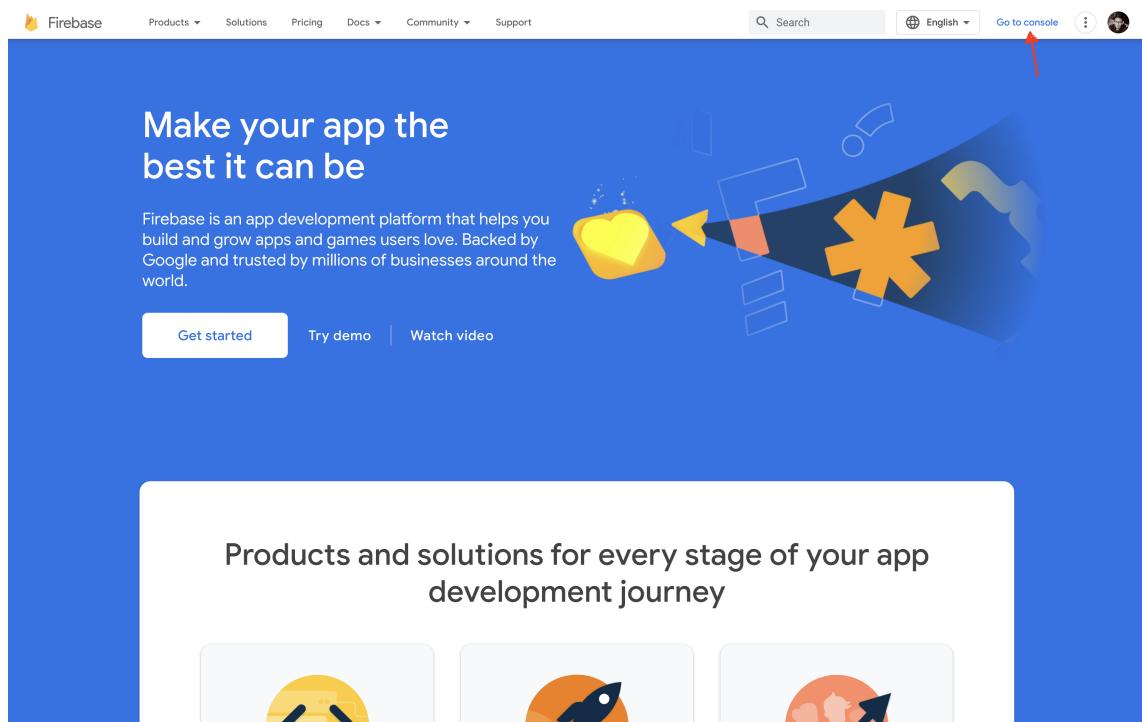
Hello, Thanks for buying the source code. Since you got your source code for “blogging website”. You need to do some things before you can access it on your browser. If you don’t follow the steps mentioned below then you won’t be able to run the site.

1. Open “blogging website” folder either in you VSCode terminal or any terminal of your choice.
2. In terminal once you are inside “blogging website” folder. Run `npm install` command

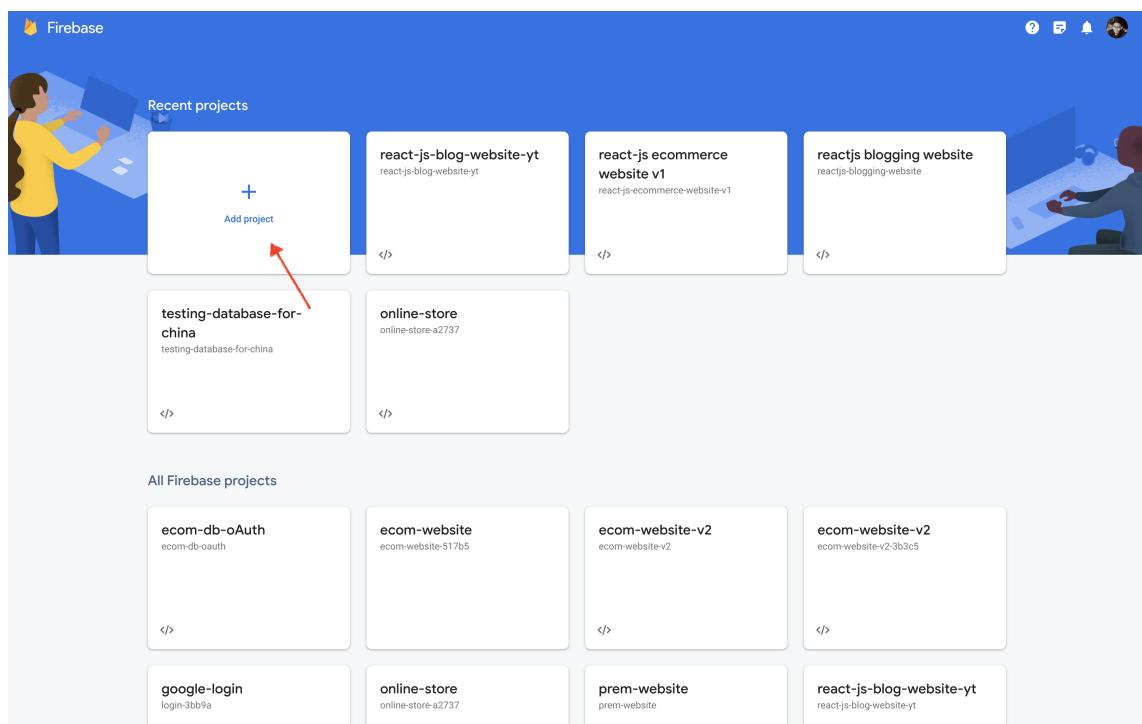
Note - If it gives error make sure you have node.js and npm installed in your system. You can check that by running `npm -v` and `node -v` cmd both cmd will give you version numbers. If you get error install nodejs from below and install it in you system. Then restart your terminal inside the folder.

Download NodeJS - <https://nodejs.org/en>

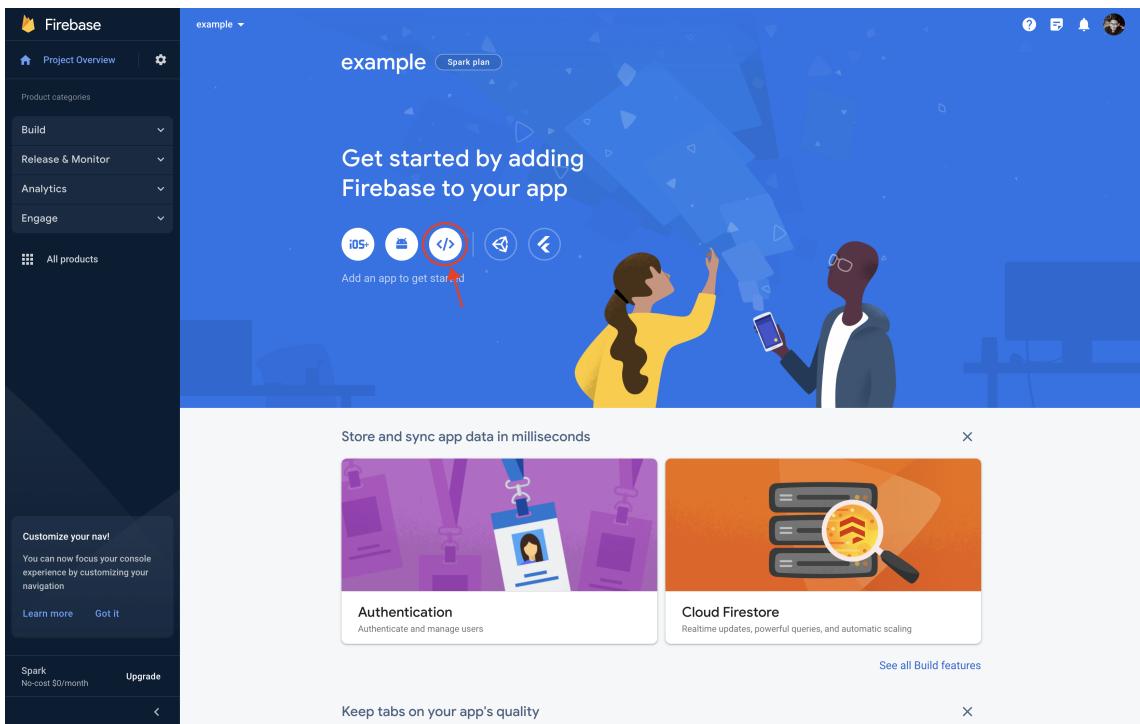
3. Once all the packages install. We need to setup our firebase. So go to <https://firebase.google.com/>
4. Log in or Create an account
5. Then if you are not in your console click on “go to console” button



6. Click on “new project”



7. Enter “Project Name” & then click “continue”
8. Turn off “google analytics” and then “create project”. It will take some time to create the project.
9. Click on “web” icon & register you app with any name.



10. Copy whole "firebaseConfig" object.

11. Then open "firebase.jsx" file inside "blogging website/src/common" folder & replace the "firebaseConfig" variable with the one that you just copied from your project.

Before

```

5 const firebaseConfig = {
6   // replace this object with your project firebaseConfig.
7 };

```

After

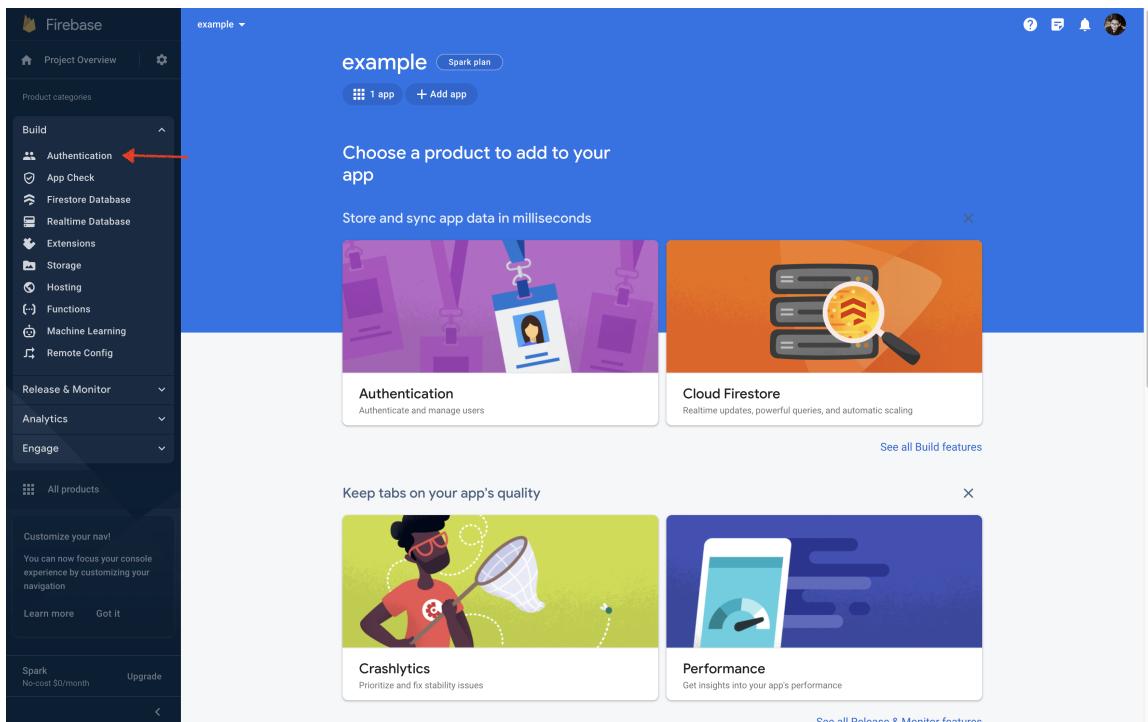
```

5 const firebaseConfig = {
6   // below values will be different for your firebase project
7   apiKey: "AIzaSyDCnU5-8tpVqnUvPkssg-w3HQwUVGrj92w",
8   authDomain: "example-944a1.firebaseio.com",
9   projectId: "example-944a1",
10  storageBucket: "example-944a1.appspot.com",
11  messagingSenderId: "407612357055",
12  appId: "1:407612357055:web:4053dc26ada836ed9f490a"
13};

```

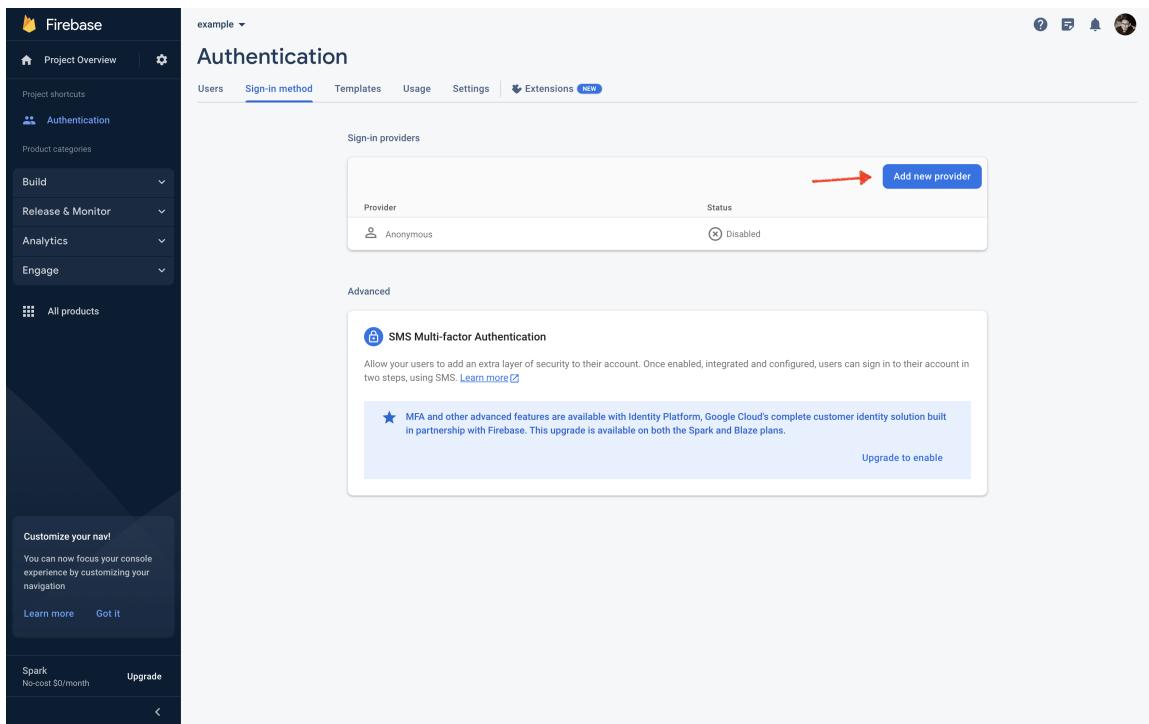
12. Now open your “server” folder in a new terminal & run `npm install` cmd.

13. Now go back to you firebase & go to “Authentication” now.

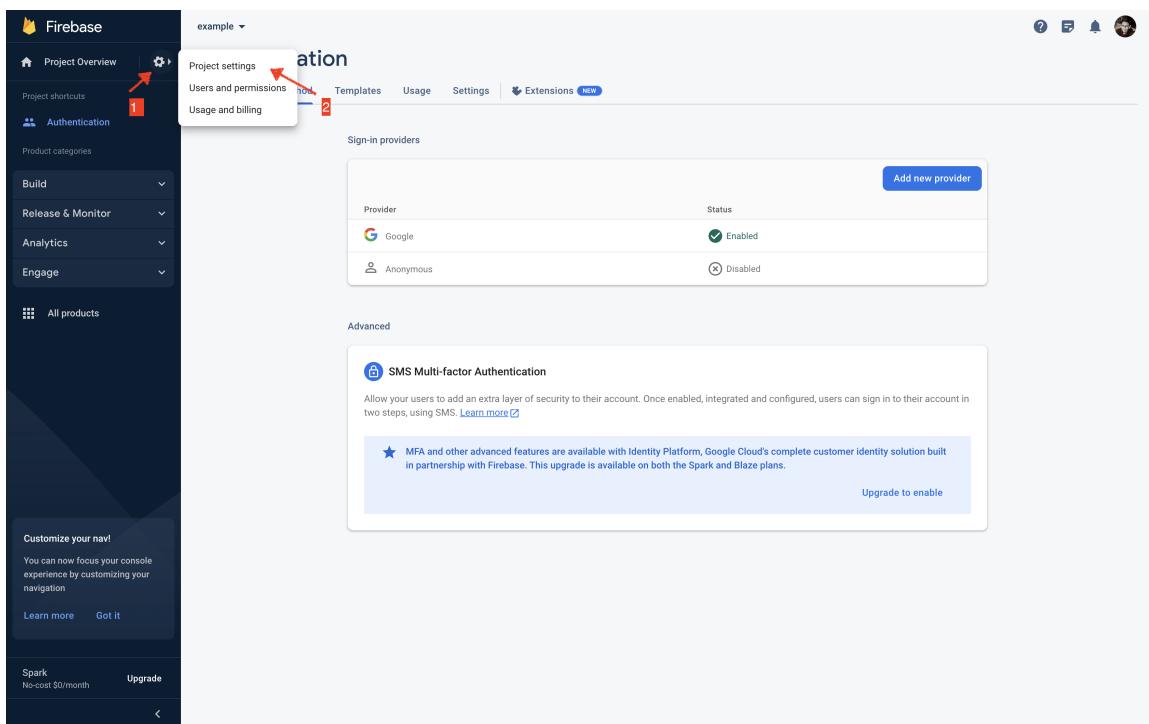


14. Click on “get Started”

15. Now click on “Add new provider”



16. Select “google” from the list of options. Turn on “Enable” and select your “email” in “Support email for project”
17. Click on save. It will add “google authentication” to your project.
18. Now, go to “project’s settings”



19. Go to “service accounts”.

Project settings

General Cloud Messaging Integrations Service accounts Data privacy Users and permissions

Your project

Project name: example

Project ID: example-944a1

Project number: 407612357055

Default GCP resource location: Not yet selected

Web API Key: AlzaSyDCnU5-8tpVqnUvPksg-w3HQwUVGrj92w

Environment

This setting customizes your project for different stages of the app lifecycle

Environment type: Unspecified

Public settings

These settings control instances of your project shown to the public

Public-facing name: project-407612357055

Support email: kunaal438@gmail.com

Your apps

Add app

Web apps

example	App nickname: example
---------	-----------------------

20. Then “generate new private key”. It will download a JSON file in your system.

Project settings

General Cloud Messaging Integrations Service accounts Data privacy Users and permissions

Firebase Admin SDK

Legacy credentials

Database secrets

All service accounts

S service accounts

Manage service account permissions

Firebase Admin SDK

Your Firebase service account can be used to authenticate multiple Firebase features, such as Database, Storage and Auth, programmatically via the unified Admin SDK. [Learn more](#)

Firebase service account: firebase-adminsdk-ch5zr@example-944a1.iam.gserviceaccount.com

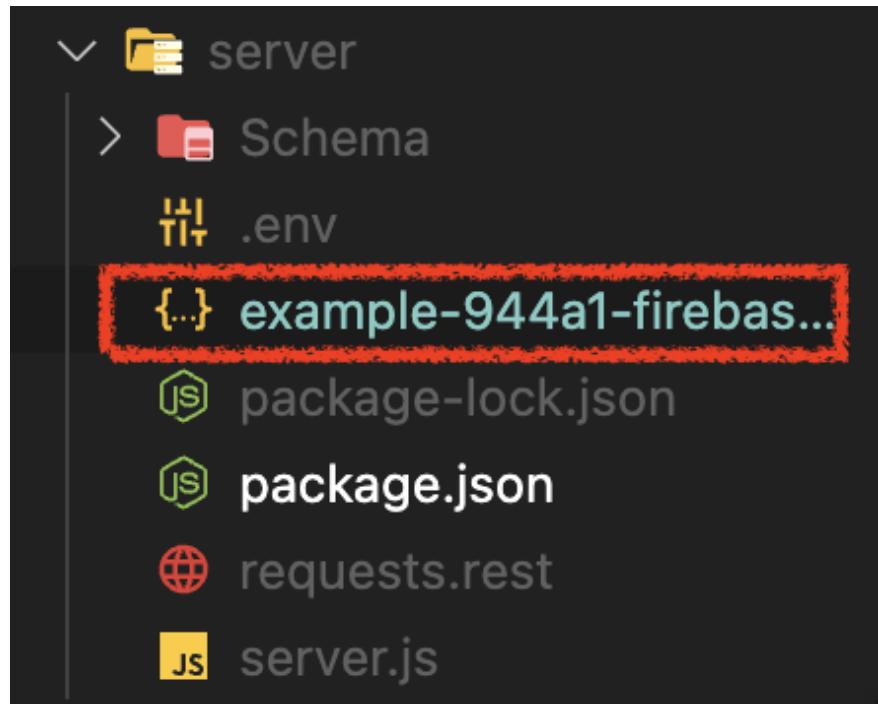
Admin SDK configuration snippet

Node.js Java Python Go

```
var admin = require("firebase-admin");
var serviceAccount = require("path/to/serviceAccountKey.json");
admin.initializeApp({
  credential: admin.credential.cert(serviceAccount)
});
```

Generate new private key

21. Cut the downloaded JSON file and paste it inside “server” folder.



22. Open “server.js” now and add your JSON file on line no. 33 inside “ ”

Before

```
33 import serviceAccount from "your firebase generated file.json" assert { type: "json" };
```

After

```
33 import serviceAccount from "./example-944a1-firebase-adminsdk-ch5zr-3cb9422ea8.json" assert { type: "json" };
```

23. Your Firebase is connected. Now let's connect MongoDB

24. Go to - <https://www.mongodb.com/>

25. Sign in or create an account. If you create account then jump directly to step 30.

26. Now create a “new project”

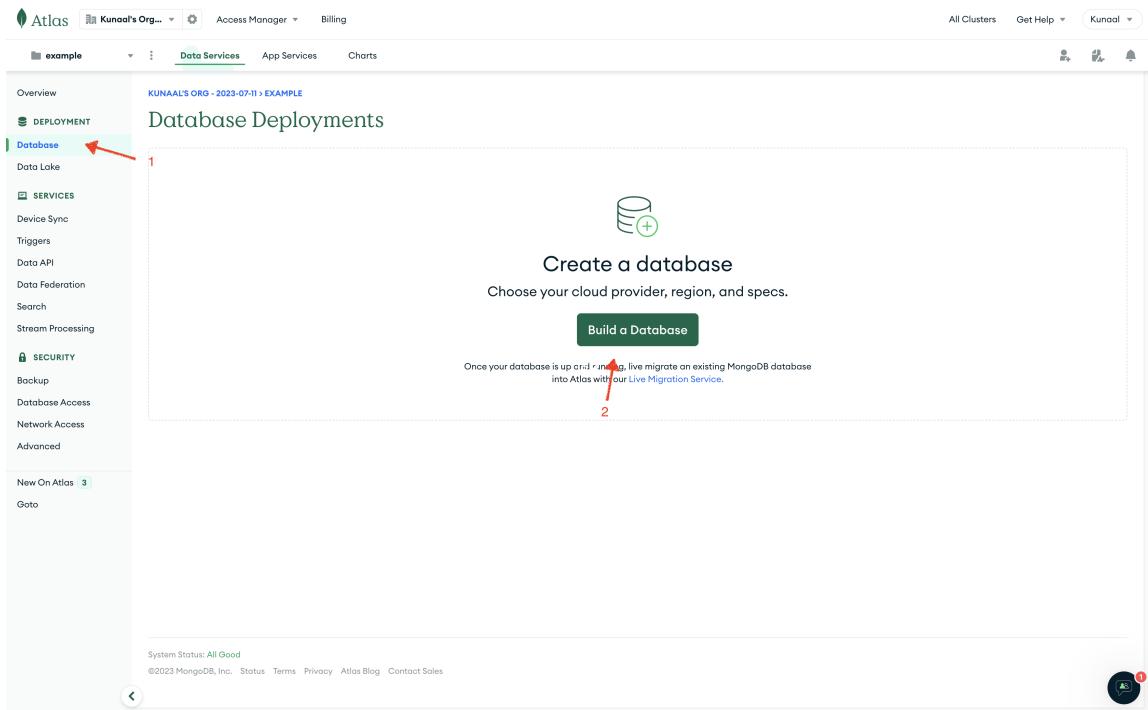
The screenshot shows the MongoDB Atlas interface. At the top, there's a navigation bar with 'Atlas', 'Kunal's Org...', 'Access Manager', and 'Billing'. Below the navigation is a search bar and a dropdown menu for 'React JS Blogging Website'. The main area is titled 'Create Deployments' and contains a 'Visualize Your Data' section with a chart and some metrics. On the left, there's a sidebar with various options like 'Data Federation', 'Search', 'Stream Processing', 'SECURITY', 'Quickstart', etc. At the bottom, there's a footer with system status and copyright information.

27. Give a project name.

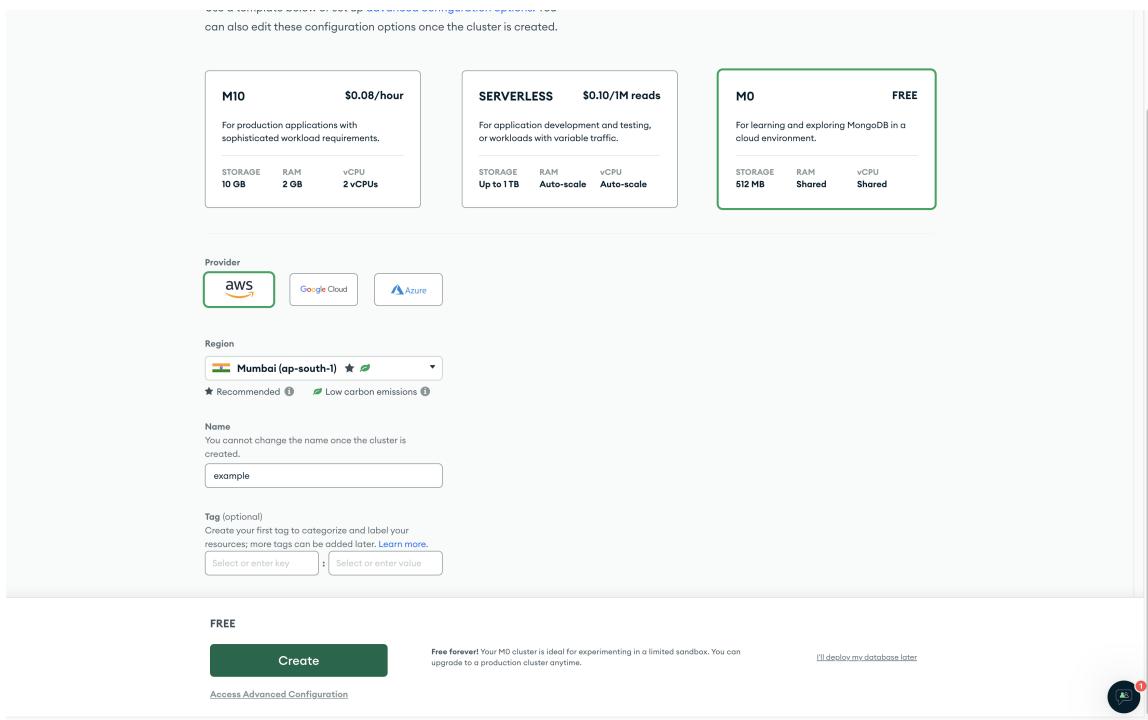
28. Since you don't have any member. Click on "create project"

The screenshot shows the 'Create a Project' page within an organization. The left sidebar includes 'Projects' (highlighted), 'Alerts', 'Activity Feed', 'Settings', 'Integrations', 'Access Manager', 'Billing', 'Support', and 'Live Migration'. The main area has sections for 'Name Your Project' (with a dropdown for 'Add Members'), 'Add Members and Set Permissions' (with a text input for email addresses), and 'Give your members access permissions below' (with a dropdown for 'Project Owner' set to 'kunal438@gmail.com [you]'). To the right, there's a 'Project Member Permissions' sidebar listing roles like 'Project Owner', 'Project Cluster Manager', etc., each with a brief description. At the bottom, there's a footer with system status and copyright information.

29. Now go to database & create a new database.



30. Select FREE tier.
31. Choose any server provider. Mine is AWS
32. In region Choose whatever region is nearest to you.
33. Give a name to cluster & Click “create”



34. You will get redirected to “Security QuickStart” page.

35. First select the “username and password” option. Under that give a username of your choice and password. If you want to autogenerated password. Click on autogenerated password on the right side of “password” field.

Note - Copy your user’s password and paste it somewhere because if its a autogenerated password you can see it only once. So copy it and paste it somewhere before creating a user.

Note - If you didn’t copy the password before creating a user. Then in the list of user’s click on “edit” icon and generate new password again and this time copy the password somewhere first before updating the password.

The screenshot shows the MongoDB Atlas interface for a project named 'EXAMPLE'. On the left sidebar, under the 'SECURITY' section, the 'Quickstart' option is selected. The main content area is titled 'Security Quickstart' and displays instructions for creating a database user. It shows two options: 'Username and Password' (which is selected and highlighted with a green border) and 'Certificate'. A note indicates that an autogenerated password was created for the first database user. Below this, there's a form to enter a 'Username' (with a placeholder 'Enter username') and a 'Password' (with a placeholder 'Enter password'). An 'Autogenerate Secure Password' button is available next to the password field, along with a 'Copy' button. At the bottom, a table shows the created user: 'kunal1438' with 'Password' as the authentication type, and edit and remove buttons. The top navigation bar includes 'Atlas', 'Kunal1's Org...', 'Access Manager', 'Billing', 'All Clusters', 'Get Help', and 'Kunal'.

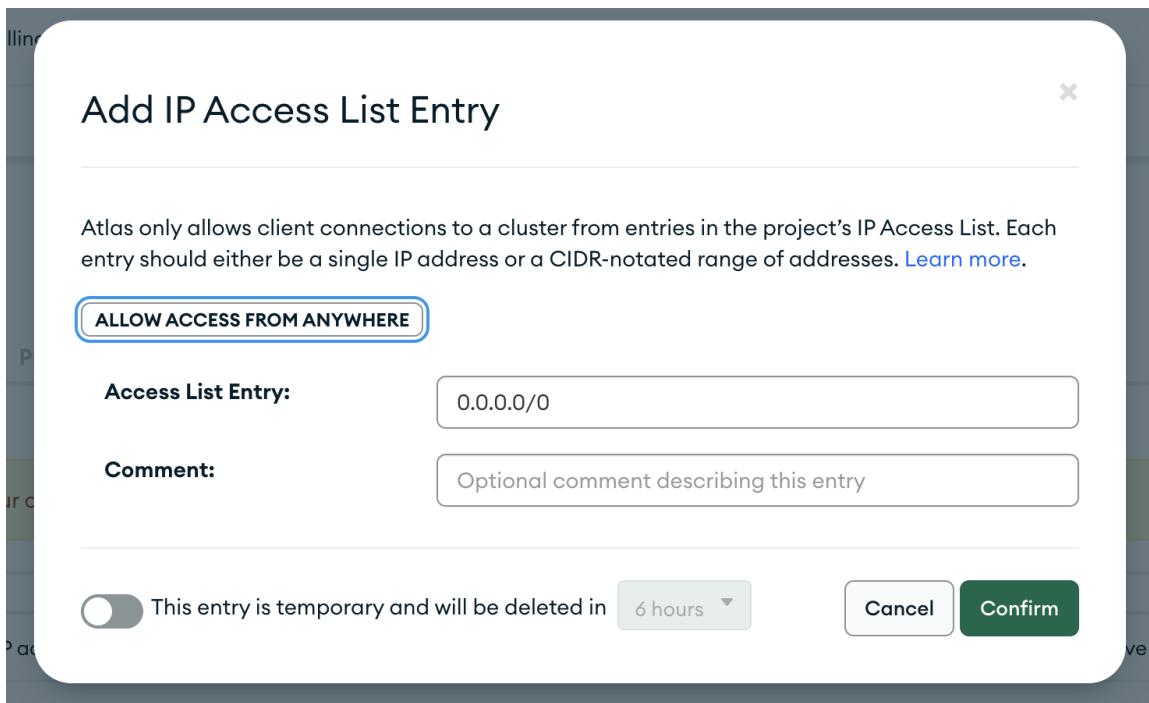
36. Now go to “Network Access” under “security”

The screenshot shows the MongoDB Atlas interface. On the left, a sidebar menu includes 'Network Access' with a red arrow pointing to it. The main content area is titled 'Security Quickstart' and displays instructions for creating a database user. It offers two authentication methods: 'Username and Password' (selected) and 'Certificate'. A note states: 'We autogenerated a username and password for your first database user in this project using your MongoDB Cloud registration information.' Below this, there's a form to enter a username ('Enter username') and a password ('Enter password'). An 'Autogenerate Secure Password' button is available. A 'Create User' button is at the bottom. At the very bottom of the user creation section, there are 'Username' and 'Authentication Type' fields, showing 'kunala438' and 'Password' respectively, with 'EDIT' and 'REMOVE' buttons.

37. Click on “Add IP address”

The screenshot shows the MongoDB Atlas 'Network Access' page. The 'IP Access List' tab is selected. A red arrow points to the '+ ADD IP ADDRESS' button at the top right of the table header. The table lists one IP address: '106.221.238.212/32 (includes your current IP address)' with a comment 'My IP Address', status 'Active', and edit/delete buttons.

38. Then click on “Allow access from anywhere” & confirm.



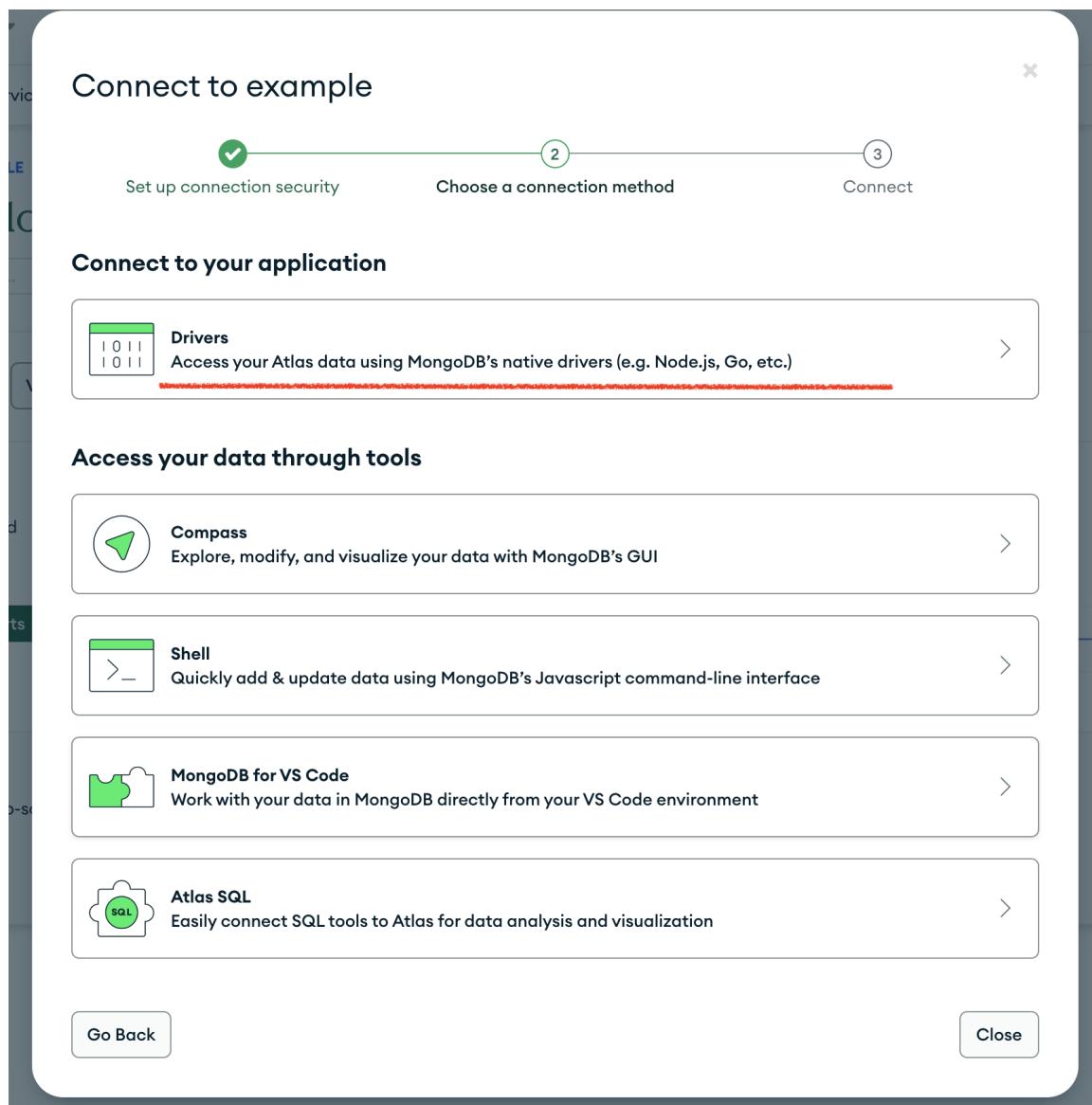
39. After that it will take some time and you will then see this in your network list. (Now you can access the database from any IP)

IP Address	Comment	Status	Actions
106.221.238.212/32 (includes your current IP address)	My IP Address	Active	<button>EDIT</button> <button>DELETE</button>
0.0.0.0/0 (includes your current IP address)		Active	<button>EDIT</button> <button>DELETE</button>

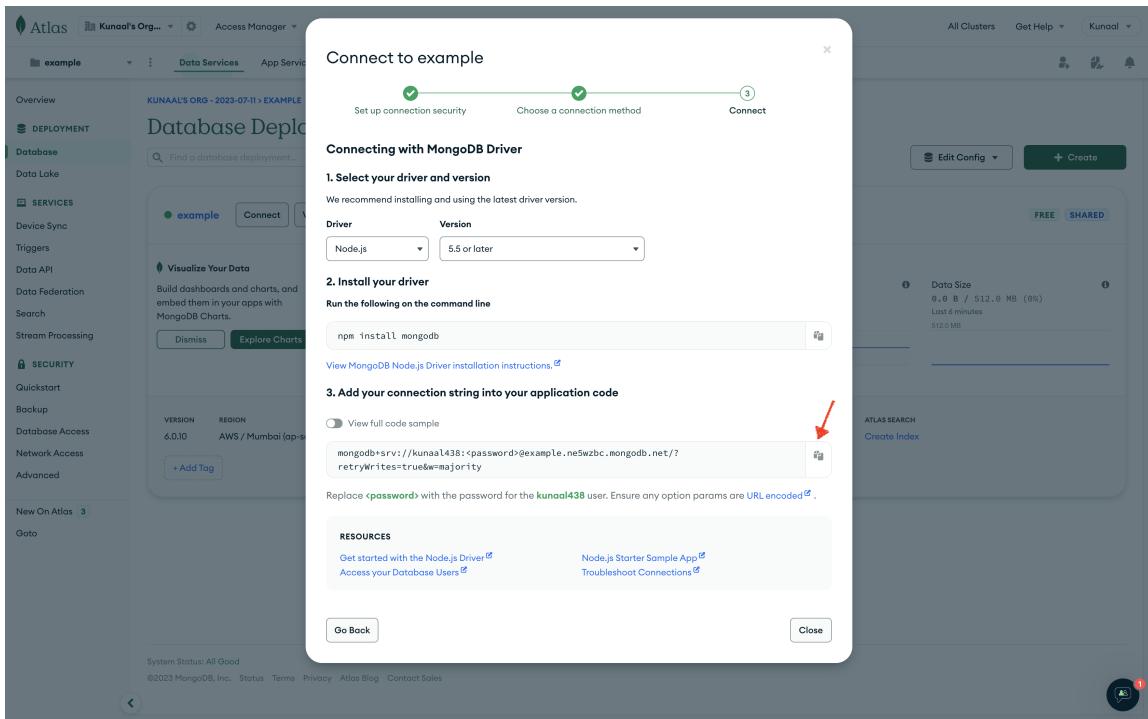
40. Now go to “database” under “deployment” & click on “connect”

The screenshot shows the MongoDB Atlas interface. The top navigation bar includes 'Atlas', 'KUNAAL'S ORG...', 'Access Manager', 'Billing', 'All Clusters', 'Get Help', and 'Kunaal'. The main menu has sections for 'Data Services' (selected), 'App Services', and 'Charts'. The left sidebar is titled 'DEPLOYMENT' and lists 'Database' (selected), 'Data Lake', 'SERVICES' (Device Sync, Triggers, Data API, Data Federation, Search, Stream Processing), 'SECURITY' (Quickstart, Backup, Database Access, Network Access, Advanced), and 'New On Atlas' (3). The main content area is titled 'Database Deployments' for 'example' cluster. It features a 'Connect' button, 'View Monitoring', 'Browse Collections', and a '...'. A 'FREE | SHARED' badge is present. A 'Visualize Your Data' button is highlighted with a red arrow labeled '2'. Below it, a message says 'Build dashboards and charts, and embed them in your opps with MongoDB Charts.' There are sections for 'Connections' (0), 'Data Size' (0.0 B / 512.0 MB), and network metrics ('In 0.0 B/s', 'Out 0.0 B/s'). At the bottom, there's a table with columns: VERSION (6.0.10), REGION (AWS / Mumbai (ap-south-1)), CLUSTER TIER (MO Sandbox (General)), TYPE (Replica Set - 3 nodes), BACKUPS (Inactive), LINKED APP SERVICES (Unable to load application data), ATLAS SQL (Connect), and ATLAS SEARCH (Create Index). A '+ Add Tag' button is also visible. The footer includes 'System Status: All Good', copyright information (©2023 MongoDB, Inc.), and links to Status, Terms, Privacy, Atlas Blog, and Contact Sales. A circular icon with a '1' is located in the bottom right corner.

41. Now click on “drivers”



42. Copy the database URL



43. Paste this URL in “.env” file inside “server” folder after “DB_LOCATION=”

```
1 DB_LOCATION=mongodb+srv://kunaal438:<password>@example.ne5wzbc.mongodb.net/?retryWrites=true&w=majority
```

44. Now replace the “<password>” from the URL with your user’s password that you have generated on step 35.

```
1 DB_LOCATION=mongodb+srv://kunaal438:sEagsRx8C0PDM0hX@example.ne5wzbc.mongodb.net/?retryWrites=true&w=majority
```

45. Now, let’s create a secret key for our JWT. So in your terminal. Run `node`

46. It will start node inside terminal. So just copy below command and paste it inside terminal and run it.

```
cmd - require('crypto').randomBytes(64).toString('hex')
```

47. This will give you a long string.

```
> require('crypto').randomBytes(64).toString('hex')
'95bcd7d810cbe50a7b4e0772191e22d720f465c3e8310c52fb46f419ed8878000058dedded5f97aab77fb710801171d97a192240d7e50fb2e851b6cf694d20b'
```

48. Copy this string and paste it inside “.env” of “server” folder after “SECRET_ACCESS_KEY=”

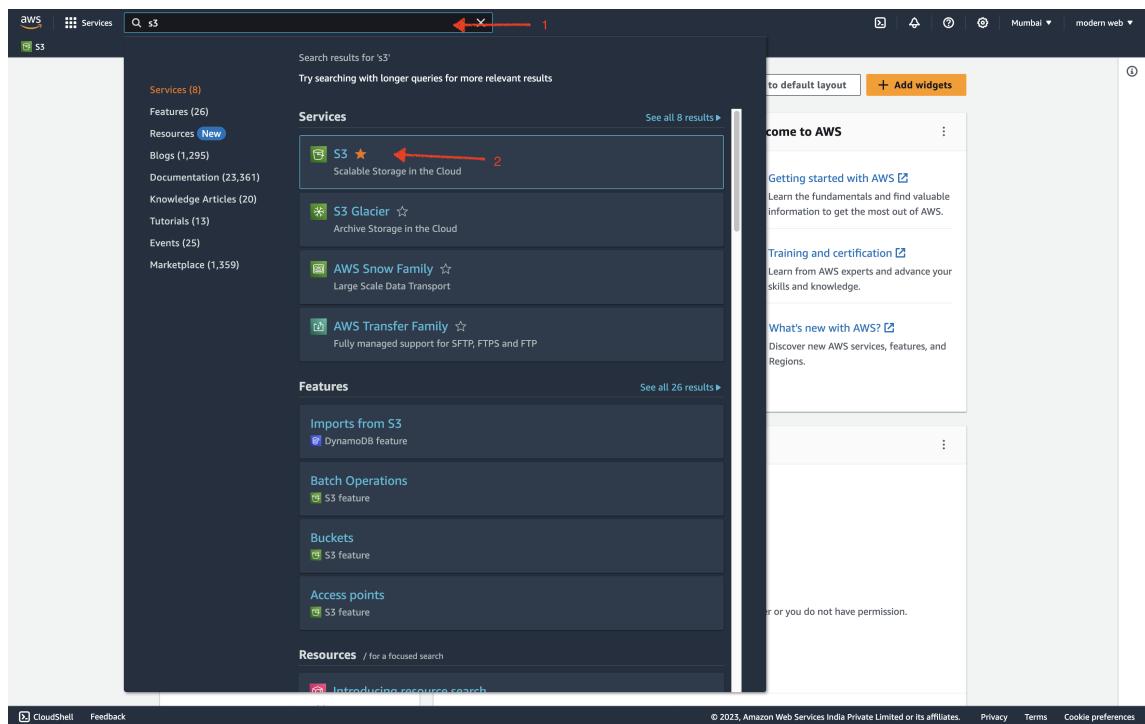
```
3 SECRET_ACCESS_KEY=95bcd7d810cbe50a7b4e0772191e22d720f465c3e8310c52fb46f419ed8878000058dedded5f97aab77fb710801171d97a
```

49. Now let’s setup AWS. Go to - <https://aws.amazon.com/>

50. Now either sign in to your AWS account or create your account there.

Note - AWS require your credit card info to create an account. Don't worry AWS is free, it will only charge you when you reach the free limit of it and since this website is just for your portfolio not a real world project, you won't exceed the free limit.

51. Now search for S3



52. Now click on “create bucket” a yellow button on right side.

53. Give a “bucket name”

54. Choose a region nearest to you.

General configuration

Bucket name

example

Bucket name must be unique within the global namespace and follow the bucket naming rules. [See rules for bucket naming.](#)

AWS Region

Asia Pacific (Mumbai) ap-south-1

Copy settings from existing bucket - *optional*

Only the bucket settings in the following configuration are copied.

[Choose bucket](#)

55. Uncheck “block all public access”

Block Public Access settings for this bucket

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. [Learn more](#)

Block all public access

Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

Block public access to buckets and objects granted through new access control lists (ACLs)

S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.

Block public access to buckets and objects granted through any access control lists (ACLs)

S3 will ignore all ACLs that grant public access to buckets and objects.

Block public access to buckets and objects granted through new public bucket or access point policies

S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.

Block public and cross-account access to buckets and objects through any public bucket or access point policies

S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.



Turning off block all public access might result in this bucket and the objects within becoming public

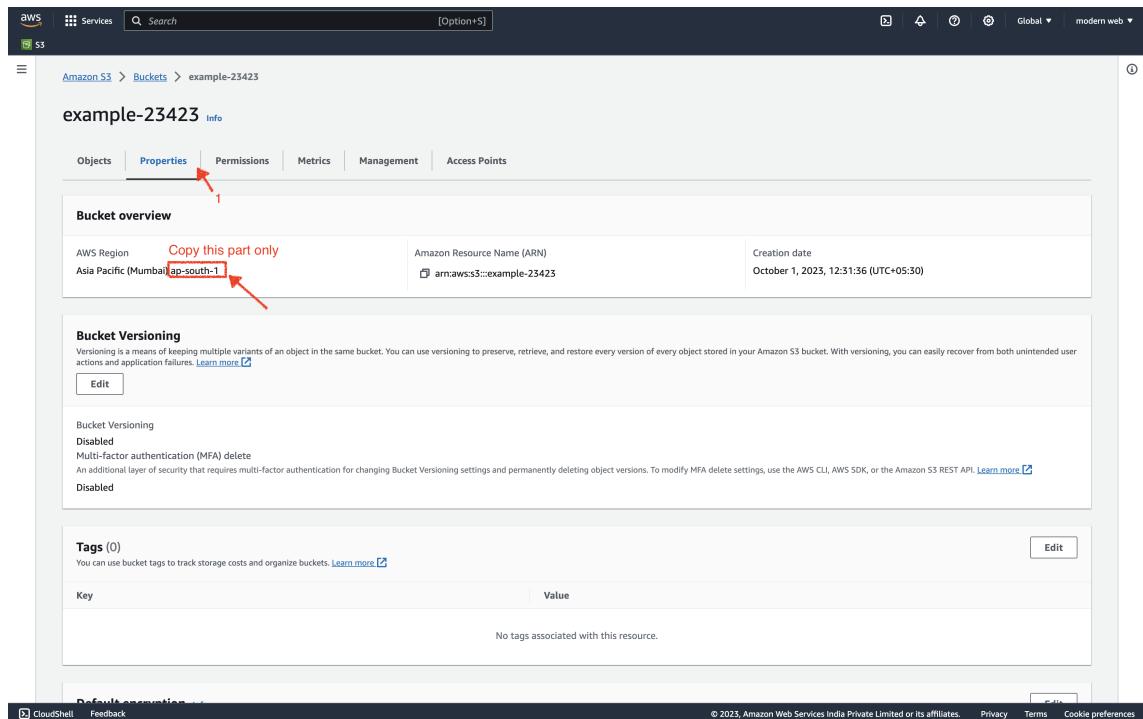
AWS recommends that you turn on block all public access, unless public access is required for specific and verified use cases such as static website hosting.

I acknowledge that the current settings might result in this bucket and the objects within becoming public.

56. And no need to change anything. Just create the bucket by clicking on the button below.

57. You will get redirected to all bucket's page. Click on the bucket name that you just create to open it.

58. Now go to projects and copy the region.



The screenshot shows the AWS S3 Bucket Properties page for 'example-23423'. The 'Properties' tab is active. In the 'Bucket overview' section, the 'AWS Region' is listed as 'Asia Pacific (Mumbai) ap-south-1'. A red box highlights this region, and a red arrow points to the 'Copy this part only' link next to it. Other details shown include the ARN 'arn:aws:s3::example-23423' and the creation date 'October 1, 2023, 12:31:36 (UTC+05:30)'.

59. And paste this inside “.env” of “server” folder after “BUCKET_REGION=”

9 **BUCKET_REGION=ap-south-1**

60. Now go to “permissions” tab. (its right after “properties” tab)

61. Scroll down and find “**Cross-origin resource sharing (CORS)**” at the end. Click on edit.

62. Now open the “cors.txt” file that you have with 2 folders in the source code. Whatever written in that file, copy that and paste it inside AWS CORS.

```

1 v [
2 v {
3 v   "AllowedHeaders": [
4 v     "/*"
5 v   ],
6 v   "AllowedMethods": [
7 v     "PUT",
8 v     "HEAD",
9 v     "GET"
10 v   ],
11 v   "AllowedOrigins": [
12 v     "/*"
13 v   ],
14 v   "ExposeHeaders": [
15 v ]
16 v }
]

```

JSON Ln 16, Col 2 Errors: 0 Warnings: 0

Cancel **Save changes**

63. Click on save changes. This will allow AWS to accept request from all IP.

64. Now scroll to top and find “Bucket Policy”, click on Edit.

65. Copy the “arn” first and then go to “policy generator”

Bucket policy
The bucket policy, written in JSON, provides access to the objects stored in the bucket. Bucket policies don't apply to objects owned by other accounts. [Learn more](#)

Policy examples [\[\]](#) Policy generator [\[\]](#)

Bucket ARN
1 [arn:aws:s3:::example-23423](#)

Policy
1

Edit statement

Select a statement
Select an existing statement in the policy or add a new statement.
+ Add new statement

66. Select “S3 Bucket Policy” in “**Select Type of Policy**”

67. Type * in “**Principal**”

68. In “Actions”, Select “GetObject” and “PutObject”
69. Now paste the “arn” you copied on step 65 to ARN field.
70. Then click on “add statement”

Step 1: Select Policy Type

A Policy is a container for permissions. The different types of policies you can create are an [IAM Policy](#), an [S3 Bucket Policy](#), an [SNS Topic Policy](#), a [VPC Endpoint Policy](#), and an [SQS Queue Policy](#).

Select Type of Policy

Step 2: Add Statement(s)

A statement is the formal description of a single permission. See [a description of elements](#) that you can use in statements.

Effect Allow Deny

Principal Use a comma to separate multiple values.

AWS Service All Services ('*')

Actions All Actions ('*')

Amazon Resource Name (ARN) ARN should follow the following format: arn:aws:s3:::\${BucketName}/\${KeyName}. Use a comma to separate multiple values.

Add Conditions (Optional)

71. Click on “generate policy” & copy the text its giving you.
72. Paste those text inside the “Bucket Policy” editor

The screenshot shows the AWS S3 Bucket Policy editor. The policy JSON is as follows:

```

1  {
2    "Id": "Policy1696144327221",
3    "Version": "2012-10-17",
4    "Statement": [
5      {
6        "Sid": "Stmt1696144315905",
7        "Action": [
8          "s3:GetObject",
9          "s3:PutObject"
10        ],
11        "Effect": "Allow",
12        "Resource": "arn:aws:s3:::example-23423",
13        "Principal": "*"
14      }
15    ]
16  }
  
```

On the right side, there is an "Edit statement" panel with the heading "Select a statement". It contains the text "Select an existing statement in the policy or add a new statement." and a button "+ Add new statement".

73. Now add “/* “ at the end of “Resources” of the policy.

12

"Resource": "arn:aws:s3:::example-23423/*",

74. Click on “save changes” now to save the policy.
75. All the Bucket setting is done. Now we need keys to connect our server with AWS.
So go to search and search for “IAM”

The screenshot shows the AWS search interface with the query 'iam' entered in the search bar. The results are categorized into Services, Features, and Resources.

- Services:**
 - IAM: Manage access to AWS resources
 - IAM Identity Center: Manage workforce user access to multiple AWS accounts and cloud applications
 - Resource Access Manager: Share AWS resources with other accounts or AWS Organizations
 - AWS App Mesh: Easily monitor and control microservices
- Features:**
 - Groups: IAM feature
 - Roles: IAM feature
 - Policies: IAM feature
 - Roles Anywhere: IAM feature
- Resources:** / for a focused search

76. Now go to “policies”

Identity and Access Management (IAM)

X

 Search IAM

Dashboard

▼ Access management

User groups

Users

Roles

Policies 

Identity providers

Account settings

▼ Access reports

77. Click on create policy. (a blue color button on the right side)
78. Under “select services” select S3. (You can search S3 there and click it)
79. Under “Actions allowed”. Search for “GetObject” & “PutObject” and check both of them.
80. Under Resources, choose “specific” and click on “Add Arn”

▼ Resources

Specify resource ARNs for these actions.

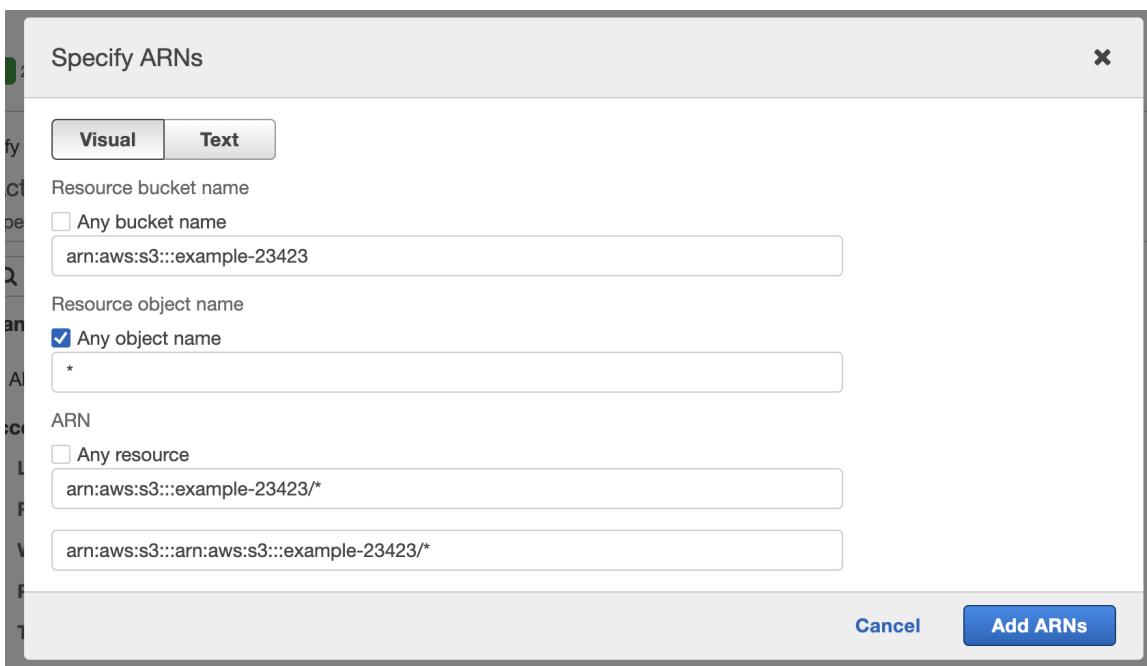
[Copy Sign-in URL to clipboard](#)

object ⓘ

⚠ Specified object resource ARN for the **PutObjectRetention** and [32 more actions](#).

[Add Arn](#) to restrict access.

81. And paste your bucket “arn” here. the same that you used to generate the policy of bucket.



82. Make sure everything else looks same like me (except the arn of your bucket) . Click on “Add Arns” now.
83. Now click “next”

Policy editor

Visual JSON Actions ▾

▼ S3 [Edit] [Delete]

Allow 2 Actions

Specify what actions can be performed on specific resources in S3.

▼ Actions allowed Switch to deny permissions ⓘ

Specify actions from the service to be allowed.

Filter Actions

Manual actions | Add actions

All S3 actions (s3:*)

Access level Expand all | Collapse all

▶ List (10)
▶ Read (Selected 1/53)
▶ Write (Selected 1/42)
▶ Permissions management (15)
▶ Tagging (10)

▼ Resources Any

Specify resource ARNs for these actions.

Specific All

object [Edit] [Delete]

arn:aws:s3:::arn:aws:s3:::example-23423/*

Add Arn to restrict access.

► Request conditions - optional Actions on resources are allowed or denied only when these conditions are met.

+ Add more permissions

Security: 0 Errors: 0 Warnings: 0 Suggestions: 0

Cancel Next

84. Give any name to the policy. To make things easy give your bucket name to the policy as well, so it will be easy for you to tell this policy is for what bucket.
85. And Click on “create policy”
86. Now go to “users” from the sidebar.

▼ Access management

User groups

Users 

Roles

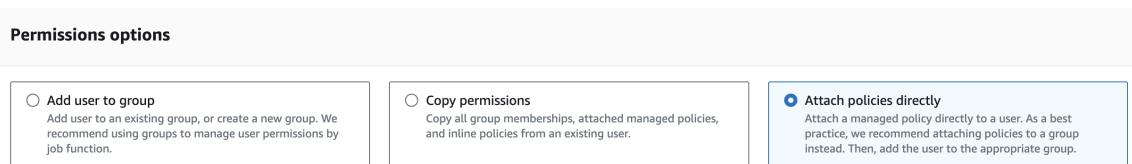
Policies

Identity providers

Account settings

▼ Access reports

87. Click on “create user” (button on the right side of the page)
88. Give any user name. Again to make things easy give your bucket name here, so it will be easy for you to tell this user can access this bucket.
89. In “**Permissions options**” Select “Attach policies directly”



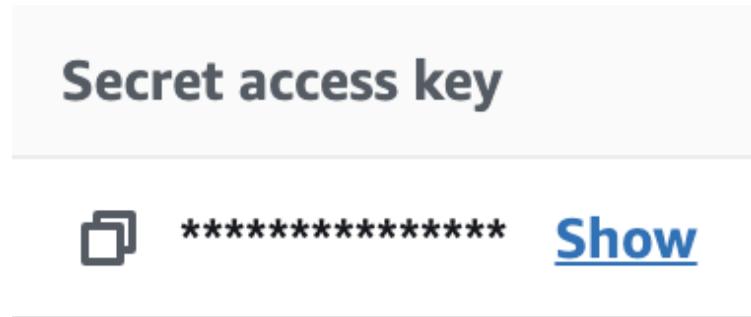
90. Now search for the policy name that you just generated on step 85 and check that.
91. Click “Next”.
92. Then click on “Create User”.
93. It will create the user.

94. Now, you will get redirected to the page where you will get all the list of users.
Search the user name that you just created and click on it to open it.
95. Under “Summary”. Click on “Create access key”

Access key 1

Create access key

96. Under use cases, select “other” and click “next”
97. Skip the tag part and click on “create access key”
98. You will get 2 keys. Now do not click on “Done” because you will get access to “secret key” only once so better copy that first.
99. Copy “secret key”



00. Now again go to “.env” of “server” folder and paste it after “AWS_SECRET_ACCESS_KEY=” (with no space)
01. After that come back to AWS and copy “**Access key**” and paste it after “AWS_ACCESS_KEY=” inside “.env” file.
Note - I can't show you the keys so therefore no screenshots here but just make sure you add no space after = for both variables and save the file.
02. Now everything is connected. You can start your server now.
03. Just one more this we have to do. Open “.env” file of “blogging website” folder. There you will find “VITE_SERVER_DOMAIN” variable.

04. Add your server URL there otherwise the frontend wont be able to make request to the server.

```
1 VITE_SERVER_DOMAIN=http://localhost:3000
```

Note - make sure you don't add "/" at the end of the localhost URL otherwise it wont work. Also if you server run on some different port rather than 3000 change the 3000 to that port number as well.

05. Now you can start your frontend and backend code on sever and access the website on browser.
06. To run frontend, open "blogging website" folder in terminal and run `npm run dev` cmd it will start the vite server for you to access the website on localhost.
07. To run the server, open "server" folder in terminal and run `npm start` cmd, it will start the node server on localhost with the PORT number 3000. mentioned in the "server.js" file.

Note - If you get error like "PORT is already in use" then in server.js file on line 20 change the PORT to any other number instead of 3000 to start your server on different PORT. Change the PORT number until you find a not running PORT. generally if you just change PORT to 3001 or 3002 this should work.

Note - Also if you change the PORT in server.js it will start the server on different PORT, so make sure you change the PORT number inside ".env" of "blogging website" where the VITE_SERVER_DOMAIN is localhost. change that URL's PORT to your current server PORT.

Now you can access your website and try using it. Adding data to it, building more features on top of that, its up to you what you want to do with that. To access website every time you have start your "vite" server and "node" server. Mean follow the step 106 and 107 every time you want to access the website.