

User Manual: Deploying Docker Container to Azure and deploy in Azure App Service

Introduction

This section shows the deployment of Docker containers to Azure as well as deploying the container as a Web app using Azure App Service.

Prerequisites

- An Azure account.
- Docker installed on your local machine.
- Azure CLI Installed on your local machine or access to Azure Portal

1. Cloning the Projects

- Cloning the server project and going to root of project.

```
git clone https://github.com/MBeattie02/code_review_tool_server.git
```

- Cloning the client project and going to root of project.

```
git clone https://github.com/MBeattie02/code_review_tool_client.git
```

2. Prepare Docker Containers

In the root of the project there is a Dockerfile to facilitate the building of the docker image. To build the image based on this Dockerfile, run the following commands:

Client application

- Building docker image for client:

```
docker build -t frontend .
```

- To run the client image locally for testing, run the following command:

```
docker run -p 3000:3000 frontend --name frontend-container
```

Server application

- Package the application in a jar to be used in the Dockerfile

```
mvn clean package
```

- Building docker image for server:

```
docker build -t serverside .
```

- To run the server image locally for testing, run the following command:

```
docker run -p 8080:8080 serverside --name server-container
```

Deploying to Azure

Deploying the application to Azure involves two steps. First, we will push each docker image to the Azure Container Registry. Once that is completed, we will create a new Web App for each application based on the Docker image created.

3. Create an Azure Resource Group

- Launch Azure CLI: Open your command-line interface and log in to your Azure account using:

```
az login
```

- Create a Resource Group: Use the Azure CLI to create a resource group, this is a container that manages multiple Azure resources using `az group create`:

```
az group create --name CSproject --location ukwest
```

--name: tag specifies the resource group name

--location: tag gives the region where it is located

4. Create an Azure Container Registry

- Create a Azure Container Registry to hold the docker images of the applications: use the `az acr create`

```
az acr create --resource-group CSproject --name CSProject --sku basic
```

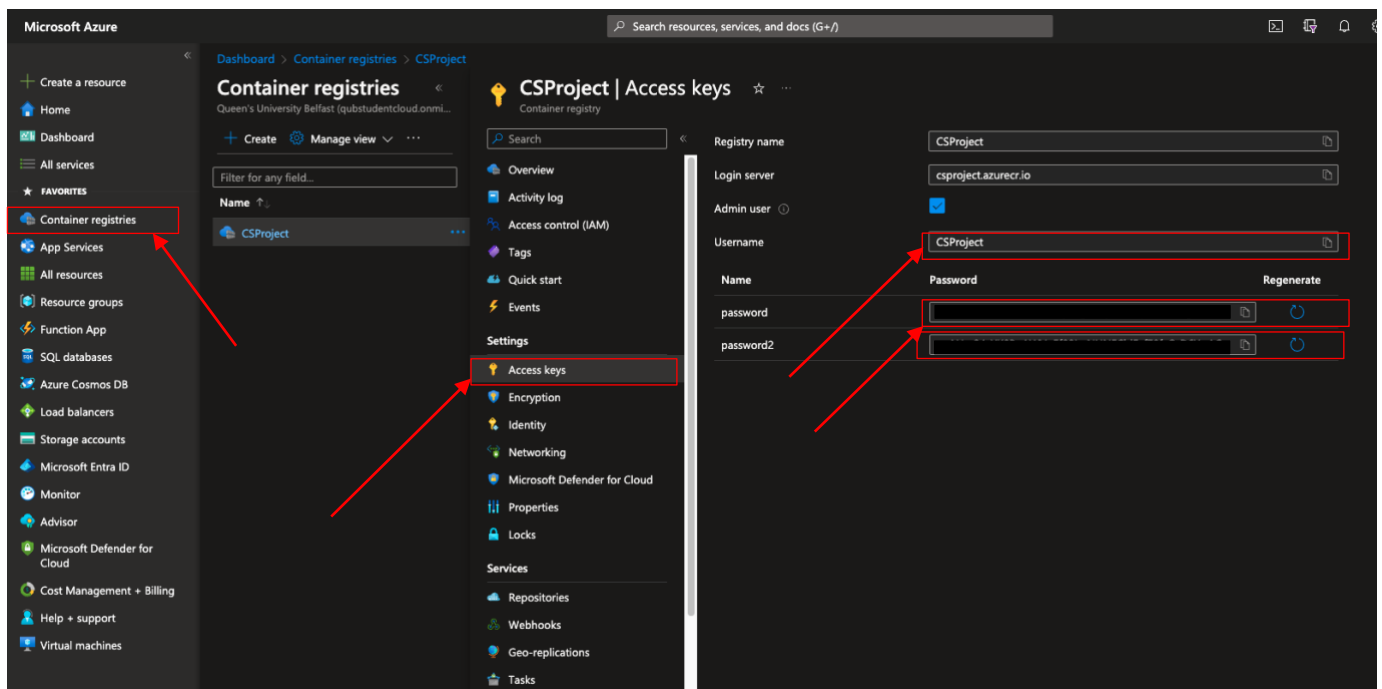
--resource-group: Specifies the name of the resource group created

--name: specifies the name of the registry (the name must be globally unique)

--sku: provides the several service tiers.

5. Obtaining Access Key from the Registry

- Docker access must be enabled in the Azure container registry. This is crucial to the deployment process as it enables you to remotely log in to the Azure container registry through the CLI and be able to push images to it.
- To do that, once the registry resources have been created successfully, open it, and locate the **Access Keys** link under the Settings section in the Container registry page.
- This will allow the retrieval of the username and password required for logging into the repository on your local machine.



6. Push Your Docker Image to Azure Container Registry

- Log in to ACR: Login requires the username and password of the users access key and the name of the login server.

```
docker login -u CSProject -p [password_of_repository] csproject.azurecr.io
```

-u: The username obtained for the container registry.

-p: The password from the container registry.

- Tag frontend Image for ACR: Tag your Docker image with the registry login server address:

```
docker tag frontend csproject.azurecr.io/frontend:latest
```

- Push the frontend Image to ACR: Deploy image to ACR using docker push:

```
docker push csproject.azurecr.io/frontend:latest
```

Expected Successful Console Output

```
matthewbeattie@MacBook-Pro clientside % docker tag frontend csproject.azurecr.io/frontend:latest
matthewbeattie@MacBook-Pro clientside % docker push csproject.azurecr.io/frontend:latest
The push refers to repository [csproject.azurecr.io/frontend]
cc570a9c081c: Pushed
c612fd39c4b2: Pushed
b697c7d27c0d: Pushed
dc07c2b04a58: Pushed
883bc20212d8: Pushed
9c2bcdc917bb: Layer already exists
0d5f5a015e5d: Layer already exists
3c777d951de2: Layer already exists
f8a91dd5fc84: Layer already exists
cb81227abde5: Layer already exists
e01a454093a0: Layer already exists
c45660adde37: Layer already exists
fe0fb3ab4a0f: Layer already exists
f1186e5061f2: Layer already exists
b2dba7477754: Layer already exists
latest: digest: sha256:fdc6213ae22a4a2fd1cf497b38630d3240236b6ce2f2c4a7667c58cf8208ace3 size: 3477
matthewbeattie@MacBook-Pro clientside %
```

- Tag server Image for ACR: Tag Docker image with the registry login server address:

```
docker tag serverside csproject.azurecr.io/serverside:latest
```

- Push the server Image to ACR: Deploy image to ACR using docker push:

```
docker push csproject.azurecr.io/serverside:latest
```

Expected Successful Console Output

```
matthewbeattie@MacBook-Pro ServerSide % docker tag serverside csproject.azurecr.io/serverside:latest
matthewbeattie@MacBook-Pro ServerSide % docker push csproject.azurecr.io/serverside:latest
The push refers to repository [csproject.azurecr.io/serverside]
7ffd770db0af: Pushed
d34077631442: Layer already exists
6be690267e47: Layer already exists
13a34b6fff78: Layer already exists
9c1b6dd6c1e6: Layer already exists
latest: digest: sha256:d65f954a8dc0bd233aa36a67646e64c86d5b238a932642bfa9d00b0f967611dd size: 1372
matthewbeattie@MacBook-Pro ServerSide %
```

The images should now be deployed to the Azure registry, this can be confirmed by checking the Azure Portal:

Frontend image deployed in Registry:

The screenshot displays the Azure Portal interface for the 'CSProject | Repositories' section. The left sidebar shows the navigation menu with 'Repositories' selected. The main content area shows the 'frontend' repository details. A table lists the tags for the repository, with the 'latest' tag highlighted. Red arrows point to the 'frontend' repository in the list and the 'latest' tag in the table.

Tags	Digest	Last modified
latest	sha256:fdc6213ae22a4a26d1c497b38630d3240236b6ce2...	04/03/2024, 16:22 GMT

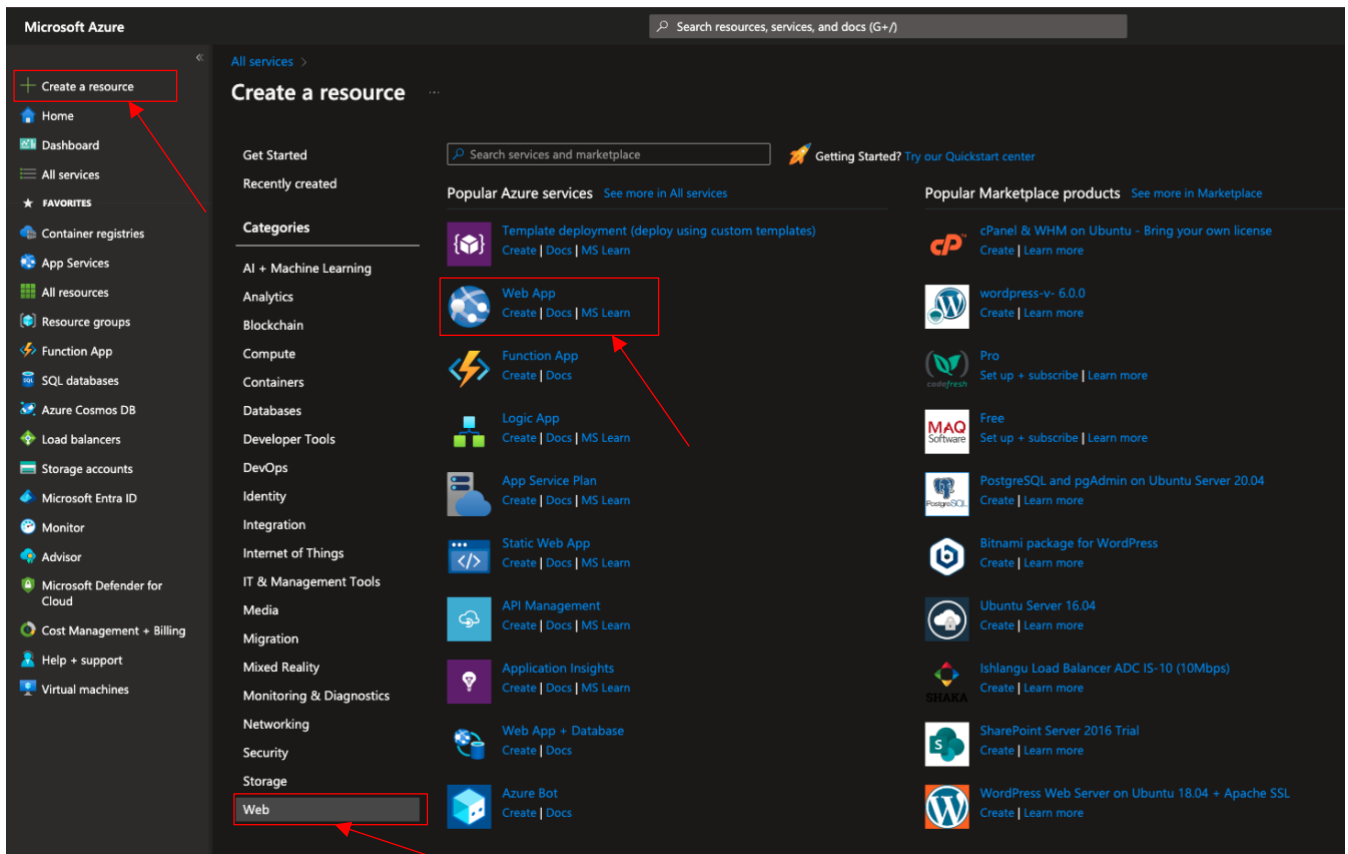
Server image deployed in Registry:

The screenshot displays the Azure Portal interface for the 'CSProject | Repositories' section. The left sidebar shows the navigation menu with 'Repositories' selected. The main content area shows the 'serverside' repository details. A table lists the tags for the repository, with the 'latest' tag highlighted. Red arrows point to the 'serverside' repository in the list and the 'latest' tag in the table.

Tags	Digest	Last modified
latest	sha256:d65f954a8dc0bd233aa36a67646e64c86d5b238a9...	04/03/2024, 16:13 GMT

7. Creating an Azure Web App for the containers

- Next, is to create an Azure Web App for each application and connect it with the container images stored in the Azure Container Registry.
- Navigate to the Azure portal homepage and click **Create a resource**.
- Then select **Web > Web App for Containers** to create a new web app service instance.



Create Web App Server

- This will then redirect to the Create Web App page.
- Select:
 - Azure subscription
 - resource group – **CSProject** as created earlier.
 - name Web App – **analyser-serverside**
 - Region – Closest datacentre to users' location
 - Pricing Plan
- Docker container should be selected by default, otherwise, select it.
 - Operating system should be Linux.

The screenshot shows the 'Create Web App' page in the Azure portal. The left sidebar contains navigation links like 'Home', 'Dashboard', 'All services', and 'App Services'. The main content area has tabs for 'Basics', 'Database', 'Docker', 'Networking', 'Monitoring', 'Tags', and 'Review + create'. The 'Basics' tab is active, showing 'Project Details' and 'Instance Details' sections. Red boxes and arrows highlight the following fields: 'Subscription' (set to 'Azure for Students'), 'Resource Group' (set to 'CSProject'), 'Name' (set to 'analyser-serverside'), 'Publish' (set to 'Docker Container'), 'Operating System' (set to 'Linux'), 'Region' (set to 'UK West'), and 'Linux Plan (UK West)' (set to 'ASP-analyserserversidegroup-83ce (F1)'). The 'Pricing plan' section shows 'Free F1 (Shared infrastructure)' as the selected plan. At the bottom, there are buttons for 'Review + create', '< Previous', and 'Next : Database >'.

Create Web App

Basics Database Docker Networking Monitoring Tags Review + create

App Service Web Apps lets you quickly build, deploy, and scale enterprise-grade web, mobile, and API apps running on any platform. Meet rigorous performance, scalability, security and compliance requirements while using a fully managed platform to perform infrastructure maintenance. [Learn more](#)

Project Details

Select a subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription Azure for Students

Resource Group CSProject [Create new](#)

Instance Details

Name analyser-serverside [.azurewebsites.net](#)

Publish ☐ Code ☒ Docker Container ☐ Static Web App

Operating System ☒ Linux ☐ Windows

Region UK West

Not finding your App Service Plan? Try a different region or select your App Service Environment.

Pricing plans

App Service plan pricing tier determines the location, features, cost and compute resources associated with your app. [Learn more](#)

Linux Plan (UK West) ASP-analyserserversidegroup-83ce (F1) [Create new](#)

Pricing plan **Free F1 (Shared infrastructure)**

[Review + create](#) [< Previous](#) [Next : Database >](#)

- Next, click on the **Docker** tab and select the image source and its respective docker image.
 - Image Source should be: Azure Container Registry
 - Azure Container registry options:
 - Registry: CSProject
 - Image: serverside
 - Tag: latest

The screenshot shows the 'Create Web App' wizard in the Azure portal, specifically the 'Docker' tab. The left sidebar contains navigation links such as 'Create a resource', 'Home', 'Dashboard', 'All services', 'FAVORITES', 'Container registries', 'App Services', 'All resources', 'Resource groups', 'Function App', 'SQL databases', 'Azure Cosmos DB', 'Load balancers', 'Storage accounts', 'Microsoft Entra ID', 'Monitor', 'Advisor', 'Microsoft Defender for Cloud', 'Cost Management + Billing', 'Help + support', and 'Virtual machines'. The main content area has tabs for 'Basics', 'Database', 'Docker' (which is selected and highlighted with a red box), 'Networking', 'Monitoring', 'Tags', and 'Review + create'. Below the tabs, a description states: 'Pull container images from Azure Container Registry, Docker Hub or a private Docker repository. App Service will deploy the containerized app with your preferred dependencies to production in seconds.' The 'Options' section has a dropdown menu set to 'Single Container'. The 'Image Source' dropdown is set to 'Azure Container Registry' and is highlighted with a red box. Below this, the 'Azure container registry options' section includes three dropdown menus: 'Registry' set to 'CSProject', 'Image' set to 'serverside', and 'Tag' set to 'latest'. Each of these three dropdowns is highlighted with a red box, and red arrows point from the 'Image Source' box to each of them. The 'Startup Command' field is empty. At the bottom, the 'Review + create' button is highlighted with a red box, and red arrows point from it to the 'Registry', 'Image', and 'Tag' dropdowns. Navigation buttons '< Previous' and 'Next : Networking >' are also visible.

- Click **Review + Create** and you will be redirected to a page where you can review the web app details. If this is correct, click **Create** to set up a new Azure web app.

Create Web App Client

- This will then redirect to the Create Web App page.
- Select:
 - Azure subscription
 - resource group – **CSProject** as created earlier.
 - name Web App – **static-code-analyser**
 - Region
 - Pricing Plan
- Docker container should be selected by default, otherwise, select it.
 - Operating system should be Linux.

The screenshot shows the 'Create Web App' page in the Azure portal. The left sidebar contains navigation links such as 'Create a resource', 'Home', 'Dashboard', 'All services', and 'FAVORITES'. The main content area is titled 'Create Web App' and includes tabs for 'Basics', 'Database', 'Docker', 'Networking', 'Monitoring', 'Tags', and 'Review + create'. The 'Basics' tab is active, showing 'Project Details' and 'Instance Details' sections. Red boxes and arrows highlight the following fields: 'Subscription' (set to 'Azure for Students'), 'Resource Group' (set to 'CSProject'), 'Name' (set to 'static-code--analyser'), 'Publish' (set to 'Docker Container'), 'Operating System' (set to 'Linux'), 'Region' (set to 'UK West'), and 'Linux Plan (UK West)' (set to 'ASP-analyserserversidegroup-83ce (F1)'). The 'Pricing plan' section shows 'Free F1 (Shared infrastructure)'. At the bottom, there are buttons for 'Review + create', '< Previous', and 'Next : Database >'.

Home > App Services >

Create Web App

Basics Database Docker Networking Monitoring Tags Review + create

App Service Web Apps lets you quickly build, deploy, and scale enterprise-grade web, mobile, and API apps running on any platform. Meet rigorous performance, scalability, security and compliance requirements while using a fully managed platform to perform infrastructure maintenance. [Learn more](#)

Project Details

Select a subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * ⓘ Azure for Students

Resource Group * ⓘ CSProject [Create new](#)

Instance Details

Name * static-code--analyser .azurewebsites.net

Publish * ☐ Code ☒ Docker Container ☐ Static Web App

Operating System * ☒ Linux ☐ Windows

Region * UK West

ⓘ Not finding your App Service Plan? Try a different region or select your App Service Environment.

Pricing plans

App Service plan pricing tier determines the location, features, cost and compute resources associated with your app. [Learn more](#)

Linux Plan (UK West) * ⓘ ASP-analyserserversidegroup-83ce (F1) [Create new](#)

Pricing plan Free F1 (Shared infrastructure)

[Review + create](#) < Previous Next : Database >

- Next, click on the **Docker** tab and select the image source and its respective docker image.
 - Image Source should be: Azure Container Registry
 - Azure Container registry options:
 - Registry: CSProject
 - Image: serverside
 - Tag: latest
- Click **Review + Create** and you will be redirected to a page where you can review the web app details. If this is correct, click **Create** to set up a new Azure web app.

Home > App Services >

Create Web App

Basics Database **Docker** Networking Monitoring Tags Review + create

Pull container images from Azure Container Registry, Docker Hub or a private Docker repository. App Service will deploy the containerized app with your preferred dependencies to production in seconds.

Options: Single Container

Image Source: Azure Container Registry

Azure container registry options

Registry: CSProject

Image: frontend

Tag: latest

Startup Command

Review + create < Previous Next : Networking >

Once the process is completed, you can visit the URL generated for your application, you will be able to view the app as deployed to Azure. The URL is in this format https://<APP_NAME>.azurewebsites.net .

- Client – Main UI user interacts with.

<https://static-code-analyser.azurewebsites.net>

- Server – Deploys backend functionality called by frontend.

<https://analyser-serverside.azurewebsites.net>

Portal showing default domain assigned:

The screenshot displays the Azure portal interface for a web application named 'analyser-serverside'. The left-hand navigation pane lists various Azure services and tools. The main content area is divided into several sections:

- Essentials:** Provides key information about the web app, including its status (Stopped), location (UK West), and subscription details. A red box highlights the 'Default domain' as 'analyser-serverside.azurewebsites.net', with a red arrow pointing to it.
- Properties:** A tab showing detailed configuration for the web app, including its name, publishing model (Container), and container image.
- Domains:** A section showing the assigned default domain and an option to add a custom domain.
- Hosting:** Details the app's hosting configuration, including the App Service plan (ASP-analyserserversidegroup-B13c), operating system (Linux), and instance count (1).
- Deployment Center:** Offers options to view deployment logs and manage the deployment center.
- Application Insights:** Provides a link to enable application insights for monitoring.
- Networking:** Displays the virtual IP address and outbound IP addresses for the web app.

Cloud Database Deployment MongoDB Atlas

Creating a MongoDB Atlas database involves setting up a managed MongoDB cluster in the cloud. MongoDB Atlas is a fully managed cloud database service that provides scalable MongoDB databases on AWS, Azure, and Google Cloud. This manual will show the setup process step by step.

Prerequisites

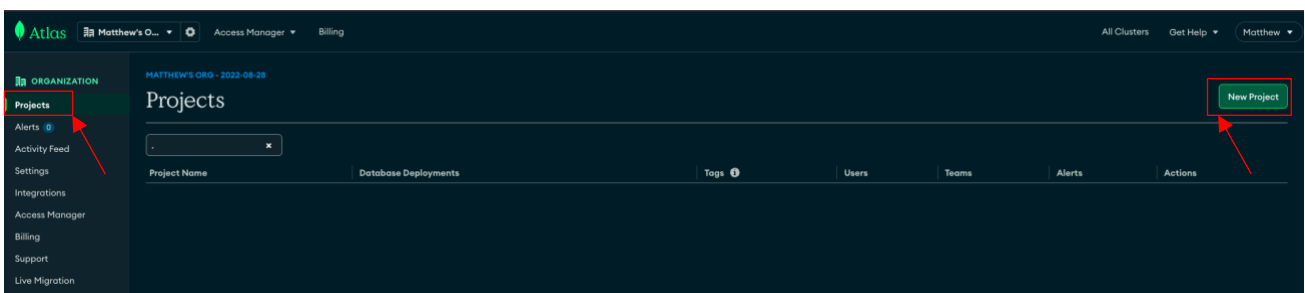
An active MongoDB Atlas account. If you don't have one, you'll need to sign up at MongoDB Atlas.

1) Sign Up and Log In

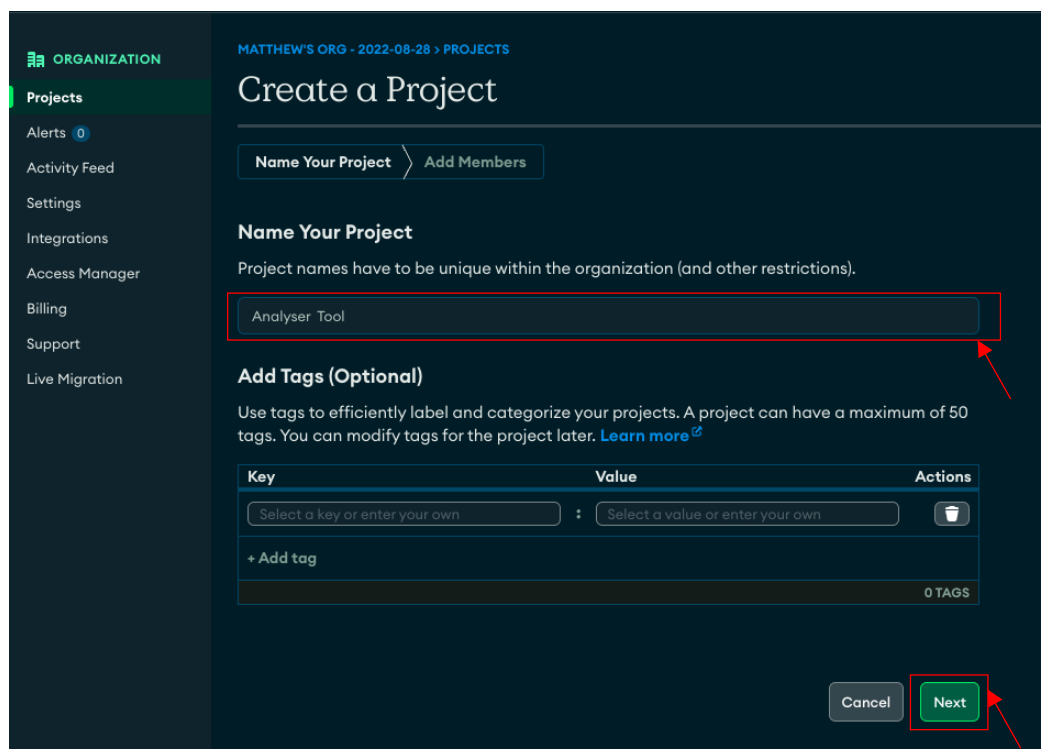
- Log In: Once your account is set up, log in to your MongoDB Atlas dashboard.

2) Step 2: Create a New Project

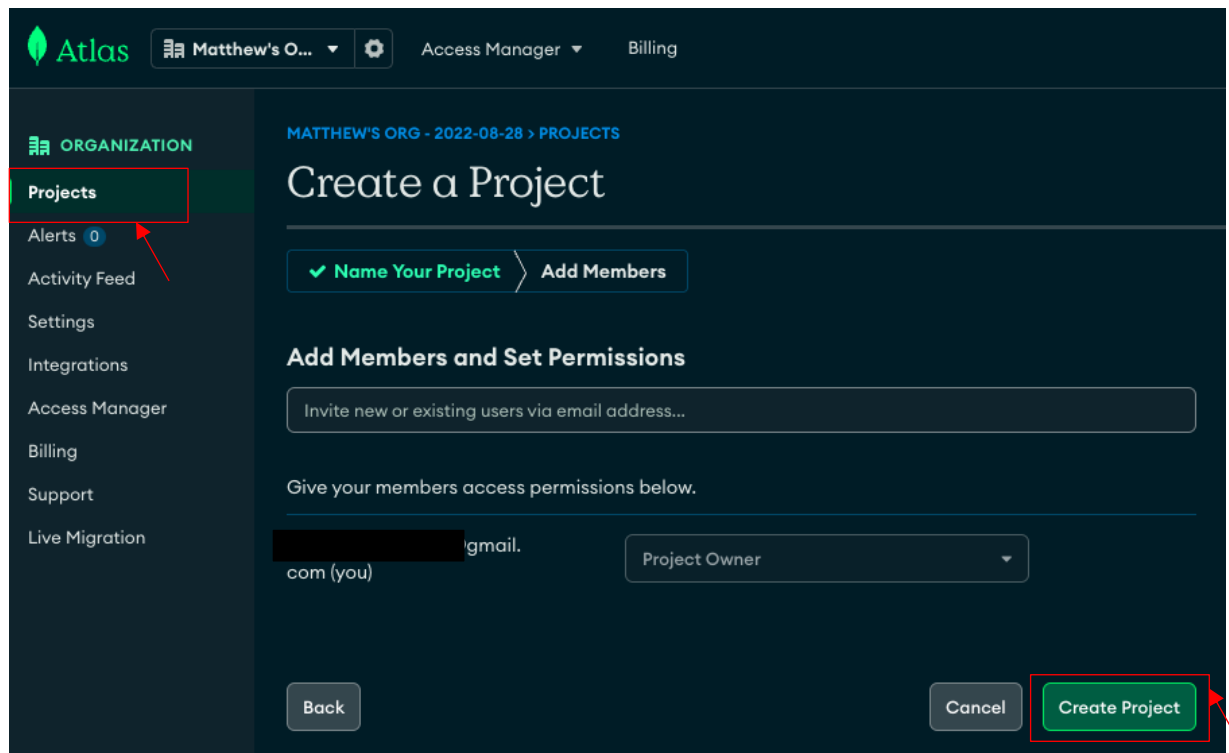
- Navigate to the Projects Page: Click the "Context" dropdown in the top left corner and select "Projects".
- Create Project: Click the "New Project" button.



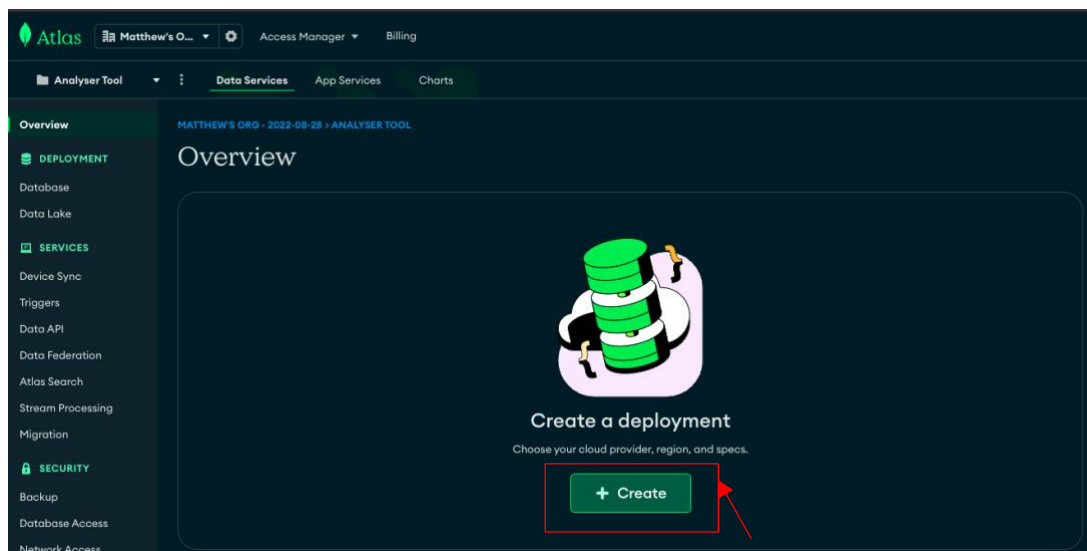
- Give project a name **Analyser Tool**, and then click "Next".



- Add Members: Optionally, you can add other members to your project by entering their email addresses. Click "Create Project" when done.



3) Build a New Cluster



- Choose a Cloud Provider and Region: After creating a project, you'll be prompted to build a cluster.
- Select your preferred cloud provider **AWS** and choose a region that is closest to your application's users for the best performance.
- Select Cluster Tier: Choose a cluster tier. For beginners or small projects, the free tier (M0) is sufficient. Click "Create Cluster" after making your selection.

MongoDB.

Deploy your database

Use a template below or set up [advanced configuration options](#). You can also edit these configuration options once the cluster is created.

M10 **\$0.09/hour**
For production applications with sophisticated workload requirements.

STORAGE	RAM	vCPU
10 GB	2 GB	2 vCPUs

SERVERLESS **\$0.11/1M reads**
For application development and testing, or workloads with variable traffic.

STORAGE	RAM	vCPU
Up to 1 TB	Auto-scale	Auto-scale

M0 **FREE**
For learning and exploring MongoDB in a cloud environment.

STORAGE	RAM	vCPU
512 MB	Shared	Shared

Provider

Region

Ireland (eu-west-1) ★

★ Recommended ⓘ Low carbon emissions ⓘ

Name

You cannot change the name once the cluster is created.

Cluster0

FREE

Create

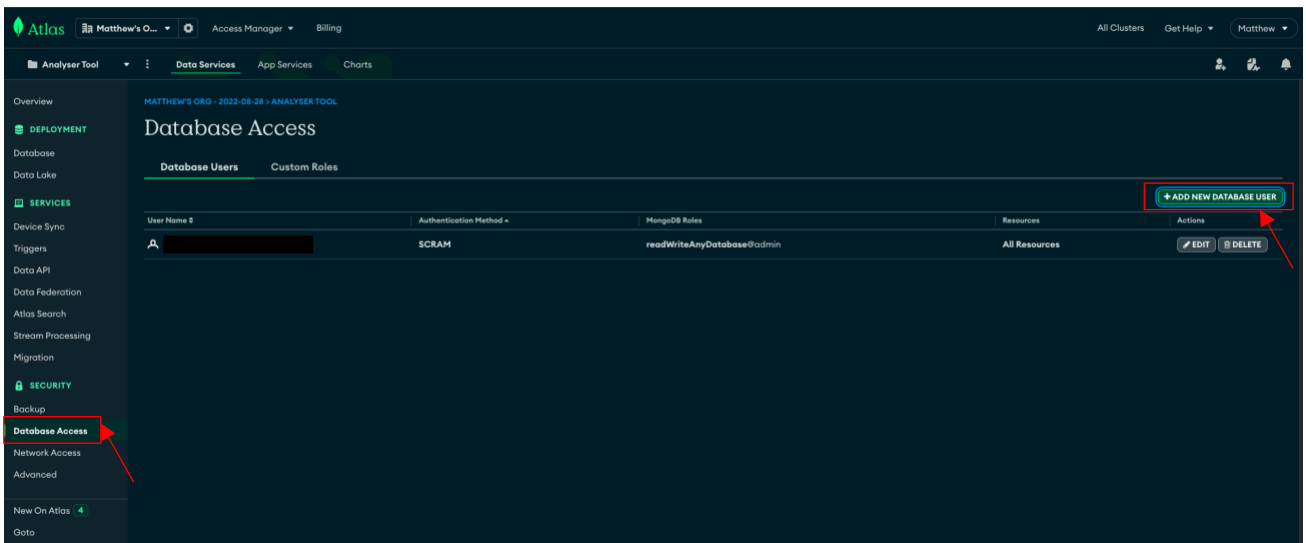
Free forever! Your M0 cluster is ideal for experimenting in a limited sandbox. You can upgrade to a production cluster anytime.

[I'll deploy my database later](#)

[Access Advanced Configuration](#)

4) Step 4: Configure Database Access

- Database Access: Navigate to the "Database Access" tab on the left-hand sidebar.



- Add New Database User: Click the "ADD NEW DATABASE USER" button. Create a user with a username and password. Assign appropriate permissions for this user.
 - "Read and write to any database"
- Save User: Click "Add User" to save the new database user.

Add New Database User

Create a database user to grant an application or user access to databases and collections in your clusters in this Atlas project. Granular access control can be configured with default privileges or custom roles. You can grant access to an Atlas project or organization using the corresponding [Access Manager](#).

Authentication Method

Password Certificate AWS IAM (MongoDB 4.4 and up) **PREVIEW** Federated Auth (MongoDB 7.0 and up)

MongoDB uses **SCRAM** as its default authentication method.

Password Authentication

e.g. new-user_31

Enter password SHOW

Autogenerate Secure Password Copy

Database User Privileges

Configure role based access control by assigning database user a mix of one built-in role, multiple custom roles, and multiple specific privileges. A user will gain access to all actions within the roles assigned to them, not just the actions those roles share in common. You must choose at least one role or privilege. [Learn more about roles.](#)

Built-in Role
Select one built-in role for this user.

Add Built In Role

Custom Roles
Select your pre-defined custom role(s). Create a custom role in the Custom Roles tab.

Specific Privileges
Select multiple privileges and what database and collection they are associated with. Leaving collection blank will grant this role for all collections in the database.

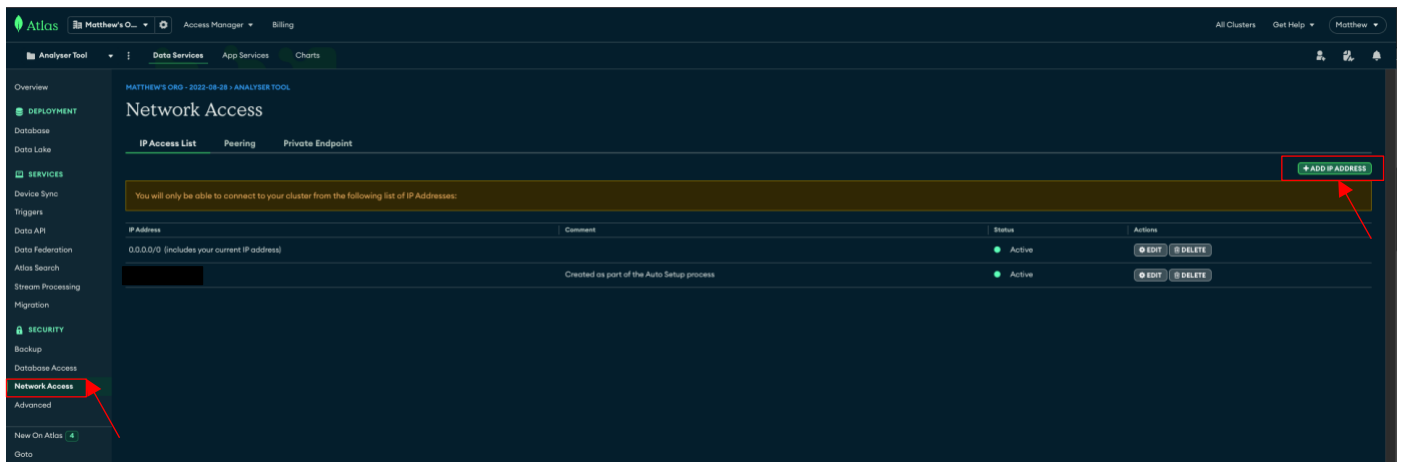
Restrict Access to Specific Clusters/Federated Database Instances/Stream Processing Instances
Enable to specify the resources this user can access. By default, all resources in this project are accessible. ☐

Temporary User
This user is temporary and will be deleted after your specified duration of 6 hours, 1 day, or 1 week. ☐

Cancel Add User

5) Step 5: Set Up Network Access

- Network Access: Navigate to the "Network Access" tab.

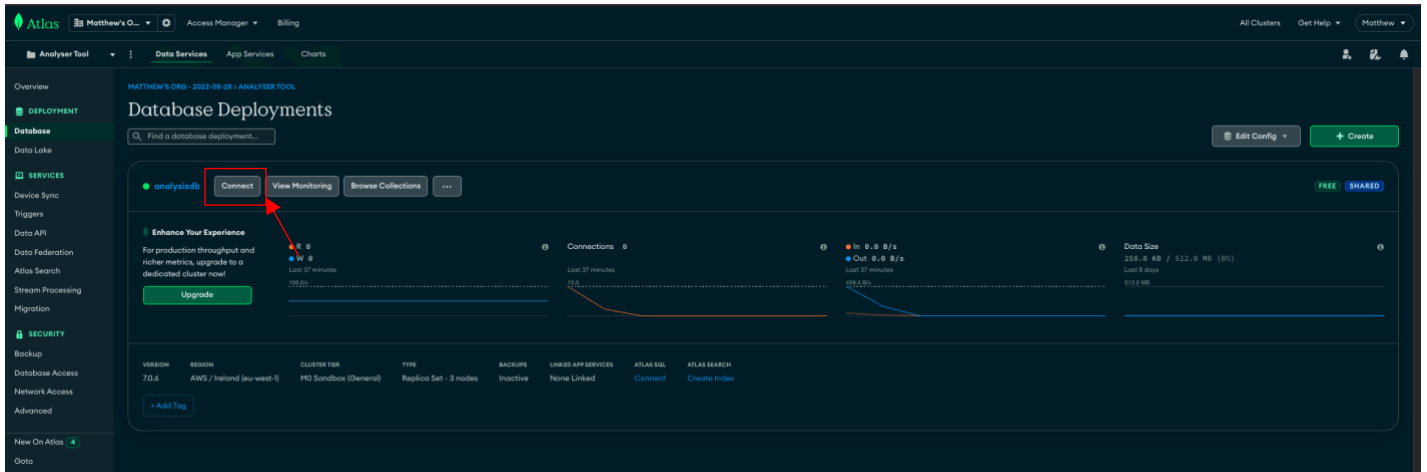


- Add IP Address: Click the "ADD IP ADDRESS" button.
- Here you can whitelist IP addresses that are allowed to connect to your database.
 - For testing purposes, you can allow access from anywhere by clicking "Allow Access from Anywhere".
- Confirm: Click "Confirm" to save your IP whitelist.

A screenshot of the 'Add IP Access List Entry' dialog box. The title is 'Add IP Access List Entry'. Below the title is a paragraph: 'Atlas only allows client connections to a cluster from entries in the project's IP Access List. Each entry should either be a single IP address or a CIDR-notated range of addresses. [Learn more.](#)'. There is a button labeled 'ADD CURRENT IP ADDRESS'. Below this are two input fields: 'Access List Entry:' with a placeholder 'Enter IP Address or CIDR Notation' and 'Comment:' with a placeholder 'Optional comment describing this entry'. At the bottom, there is a toggle switch labeled 'This entry is temporary and will be deleted in' followed by a dropdown menu showing '6 hours'. To the right of the toggle are two buttons: 'Cancel' and 'Confirm'. The 'Confirm' button is highlighted with a red box and an arrow.

6) Step 6: Connect to Your Database

- Clusters Dashboard: Go back to the "Clusters" page.
- Connect Button: Go to the created cluster and click the "Connect" button.



- This project connects by setting the mongodb uri in the applications.properties of the server codebase.

```
spring.data.mongodb.uri=mongodb+srv://[username]:[password].mongodb.net/analysisdb
```