

Atlas Forms - Quickstart

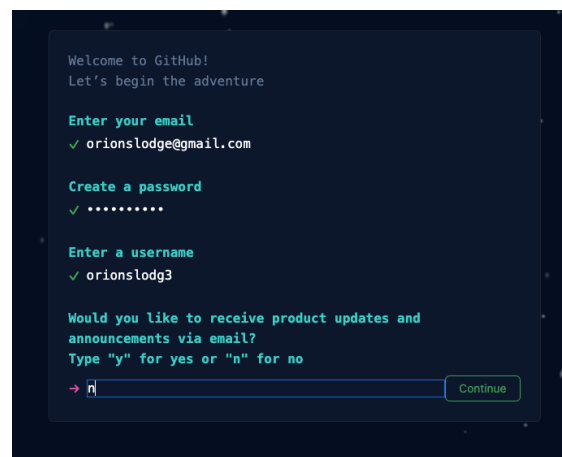
AWS re:Invent Workshop from MongoDB

Important: If we run out of time to complete all the tasks in this you can continue it later, photograph this paragraph and use the following URL to take you to an online version of this document (<http://mdb.link/AtlasForms>) you can use later. Or now if you prefer it to using this paper copy.



This set of instructions will enable you to install and host your first end-user ready application either using a supplied data set of another of your choice. It is followed by a brief introduction to customizing and managing your application.

If you do not already have one, **sign up** for an account at **GitHub** (www.github.com) - this will allow you to manage and edit the code for this project online. It takes about a minute to do and is worth it for the animation.



In your browser open the public GitHub repository for this project (<https://github.com/mongodb-developer/AtlasForms>) then at the **top right**, select the **Fork** dropdown and then **Create a new fork**. This will make your own private copy of the code linked back to the original to allow you edit to pull and push changes from the main repository if required.

Create a new fork

A fork is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project. [View existing forks.](#)

Owner ^{*} orionslodge / Repository name ^{*} AtlasForms ✓

By default, forks are named AtlasForms is available. team repository. You can customize the name to distinguish it further.

Description (optional)
Forms Based UI for MongoDB Atlas

Copy the main branch only
Contribute back to mongodb-developer/AtlasForms by adding your own branch. [Learn more.](#)

ⓘ You are creating a fork in the orionslodge organization.

Create fork

If you do not already have a MongoDB Cloud account, sign up for one at <https://account.mongodb.com/account/register> clicking **Try Free**, *you do not need to provide any payment details*. Once you verify your email for to cloud.mongodb.com and click **Sign In**,

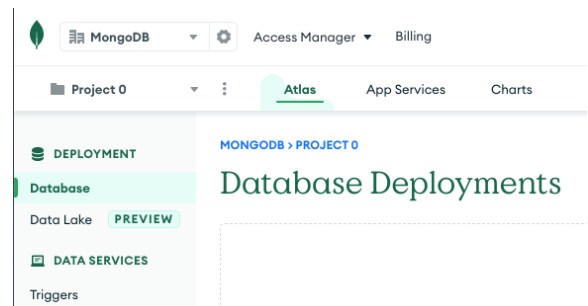
Sign up

See what Atlas is capable of for free

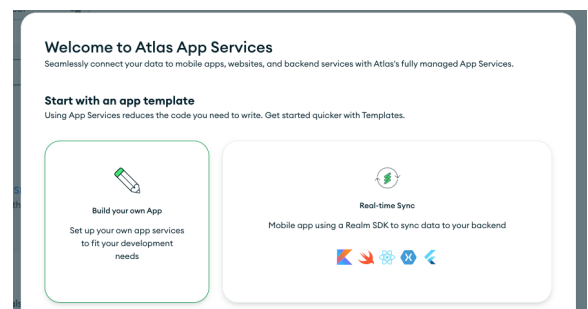
First Name*

Last Name*

If you are new to Atlas you will be offered the choice to "Build a Database" but we are going to **ignore this** and go straight to creating an application and an associated database so instead **click App Services** on the second row of the menu bar.



If this is your first time using App Services you are offered the change to "Start with an app template". Select **Build your own App** then **Next**. If it's not your first time then just **Create a New App**.



The next screen defines our application, it is Important to fill this out correctly for this workshop, do not choose your own options.

Leave the Data Source option alone.

Name the Application : **AtlasForms**

Ensure the deployment model is: **Global**.

Virginia(us-east-1) AWS

Then click **Create App Service**

1 Link your Data Source
Template Starter Applications let you build on top of data in a MongoDB Atlas cluster.
Create New M0 Atlas Cluster

2 Name your Application
This name will be used internally and cannot be changed later.
AtlasForms

3 App Deployment Model
Global • Virginia (us-east-1) • AWS | Region chosen based on your cluster's region

If you see "Welcome to Application Guides" -

Click Close Guides and any other popups.

You will then see the application console.

AtlasForms App ID: atlasforms-heing

Get Started With App Services

- REALM CLI**: Enable Local Development with the Command Line Interface.
- SDKS**: Install SDKs and get familiar with the SDK syntax.
- DOCUMENTATION**: Visit App Services documentation to learn how to build with App Services.
- DEPLOYMENT**: Connect to a GitHub account and repo to use Git.

Usage This Month

Requests	Data Transfer	Compute Runtime	Sync Runtime
0	0.00 GB	0.00 Hours	0.00 Hours

Recently Opened

No recently opened items.

We now want to start loading in some example data to our newly created highly available database cluster - this takes a little while so we will start it now. **Click Atlas** on the second row of the menu bar to take us to Database Cluster Management.

MongoDB Access Manager Billing

Project 0 Atlas **App Services**

Apps

AtlasForms App ID

NO ENVIRONMENT

Close any pop ups telling you about new features or how MongoDB can help you and then **click the three dots** shown and **choose Load Sample Dataset and confirm the popup.**

Database Deployments

AtlasCluster Connect View Monitoring Browse Collections

Enhance Your Experience

For production throughput and richer metrics, upgrade to a dedicated cluster now!

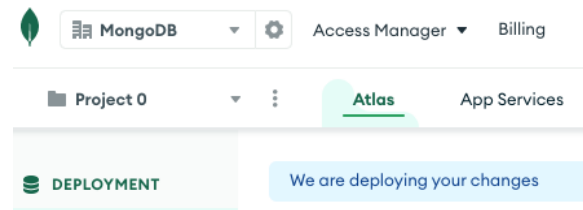
Upgrade

VERSION	REGION	CLUSTER TIER	TYPE	BACKUPS
5.0.13	AWS / Ireland (eu-west-1)	M0 Sandbox (General)	Replica Set - 3 nodes	Inactive

Load Sample Dataset

While our data imports we will go and create an application for it, **Click App Services** on the second row again to jump back to application management.

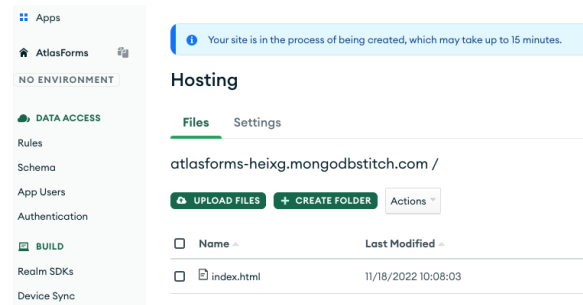
Then **click on the Atlas Forms application** at the bottom left. You can have multiple applications per project so if you have used this before you may have more than one.



Atlas Forms can be hosted using Atlas Hosting or you can host the content of the hosting/files directory anywhere else, they are static HTML, JS and CSS files.

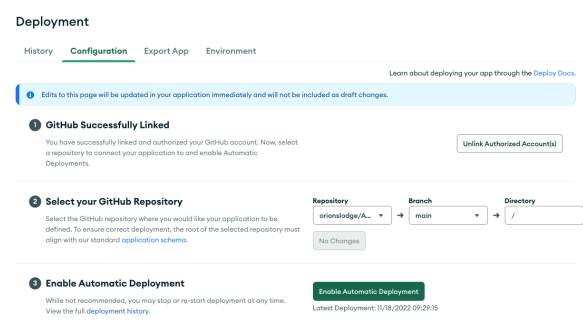
Atlas Hosting can be enabled by anyone with a paid cluster attached, if you want to use it with the free-tier, you need to add a payment method, create a 'Serverless' cluster and attach that. Once you enable hosting you can detach the cluster - you do not need to spend any money on doing so but anonymous hosting is too easy to abuse. The instructions below assume you have attached a paid cluster (in addition to the free cluster) to allow you to turn on hosting.

Now we are going to enable static hosting. This is hosting files on Cloudflare CDN to be the front-end of our application. **Click Hosting** near the bottom of the items on the left (under Manage). Then **click Enable Hosting**. Dismiss any pop ups and you should see a web address shown. **Take a note of the URL**, also **note down your Unique App Id** which is the five random characters after atlasforms in the URL.



Now we are ready to attach our App Services Application to our Github Repository and import the Atlas Forms application. To do this **click Deployment** (just above hosting on the left) then **click Configuration** (second in list of tabs within Deployment).

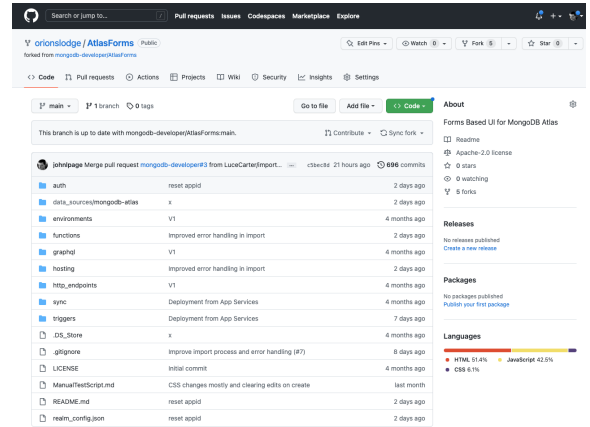
Work through the **first three steps** on the page, once you complete step 1 close the github tab that opened to let you do it. **Select**



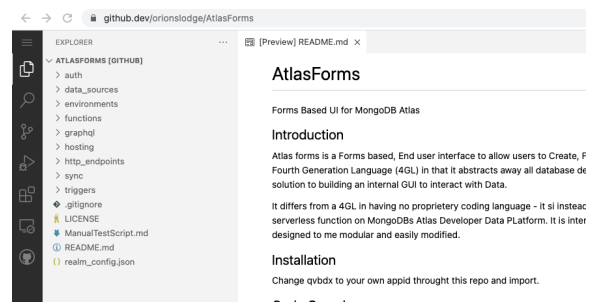
the Atlas form repo, main branch , leave the directory as is, click save and then click Enable Automatic Deployment. Confirm the pop-up by clicking Enable Automatic Deployment again.

Now when we make changes to the code in github it will be automatically deployed to App services and vice versa. It's important to change the github code here first to make the github version overwrites the defaults to return to your repo in github in a new tab. The URL will be

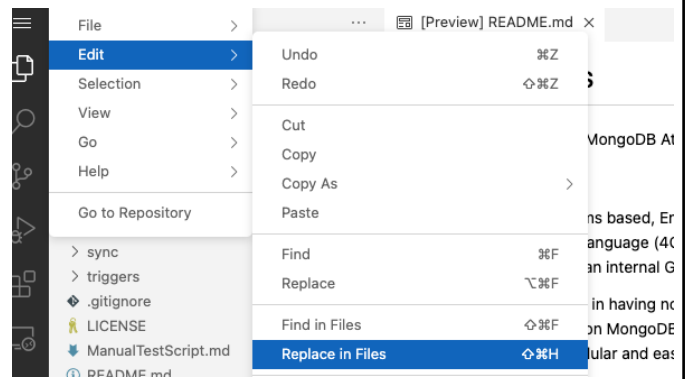
https://github.com/YOUR_USERNAME/AtlasForms



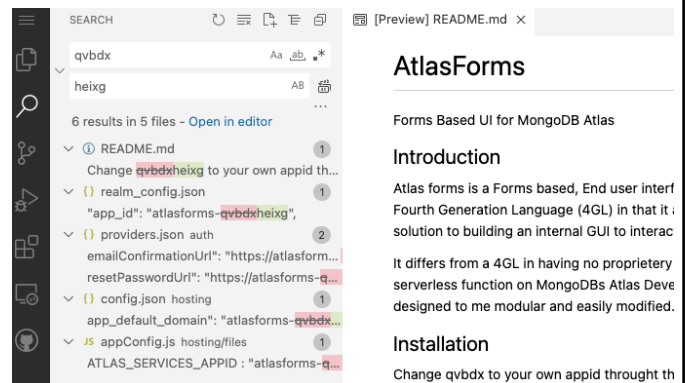
We are now going to use *github.dev* - version of Visual Studio Code built into GitHub to edit our application. To enable this just press the period (.) key on the **github** page. This will open an online IDE.



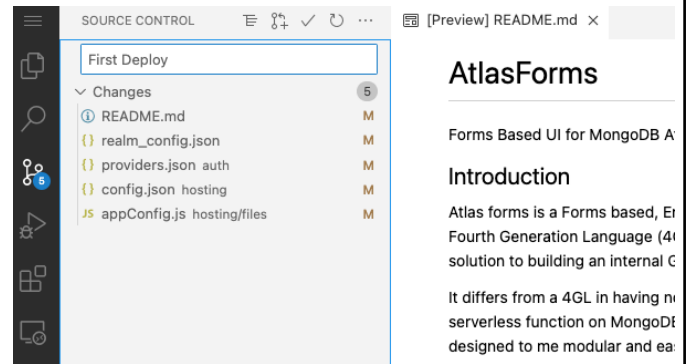
Click the **Hamburger menu** at the top left and select **Edit->Replace in files**



Now enter the string **xxxab** in the *Find* box and **your Unique App Id** in the *Replace* box then click the **replace all button** next to the replace box, confirm you want to replace 6 instances in 5 files.



Then **click the three circles on the left with the blue dot on it**, type a comment like 'Version 1' in the box then **click the check mark above**. to deploy the change.



Open the URL you noted earlier , if you get a 'Hello World' you may need to refresh as the site is still setting up. **Add yourself as a new user, acknowledge the confirmation email (you can use your phone for that) then log in** to your own application. All the code for this application is in your GitHub repo and changes you make in GitHub or the Atlas UI will reflect as soon as you commit them.

Email:

Password:

Login

New User Reset Password

As the first user, you will be logged in as a superuser, this will allow you to manage users and all the rules and data sources. We are going to add a data source so we have something to work with. By default you are seeing the user management form but that's not interesting right now as it's just you.

Atlas Forms AF_Users Create Search Clear Edit Logout

Id	Data Email	Createdate

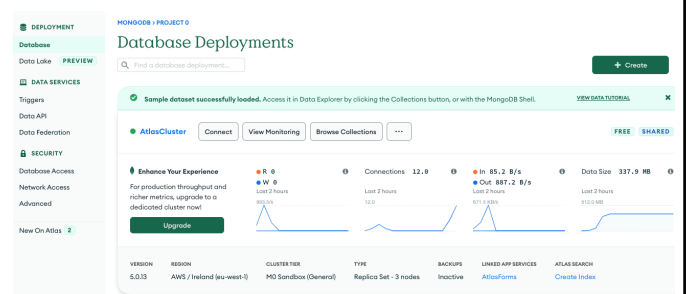
Id Type

Data Email

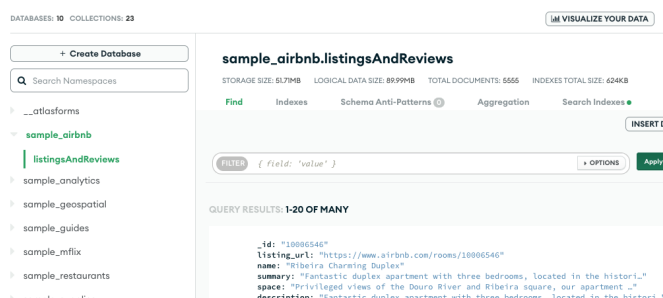
Permissions Item Permissions IsSuperUser

Custom Data Id Provider Type Createdate dd/mm/yyyy, --:--

We already loaded sample data so now we can create a frontend to allow users to interact with that data. Let us take a very brief diversion to look at the data we loaded and also to enable full text indexing of it. From cloud.mongodb.com again **click Atlas on the 2nd row** to see this view.



Click Browse Collections, then **Click sample_airbnb** then **ListingsAndReviews** - you can see the sample AirBNB rental property data we are going to build an application for, to allow you to search and manage this data. To enable Search - **click 'Search Indexes'** just above the sample document - it has a green spot on it.



DATABASES: 10 COLLECTIONS: 23 [VISUALIZE YOUR DATA]

+ Create Database

Search Namespaces

- ...atlasforms
- sample_airbnb
 - listingsAndReviews**
 - sample_analytics
 - sample_geospatial
 - sample_guides
 - sample_mflix
 - sample_restaurants

sample_airbnb.listingsAndReviews

STORAGE SIZE: 51.7MB LOGICAL DATA SIZE: 89.99MB TOTAL DOCUMENTS: 5555 INDEXES TOTAL SIZE: 624KB

Find Indexes Schema Anti-Patterns Aggregation Search Indexes

INSERT DOCUMENT

FILTER { field: "value" } OPTIONS Apply

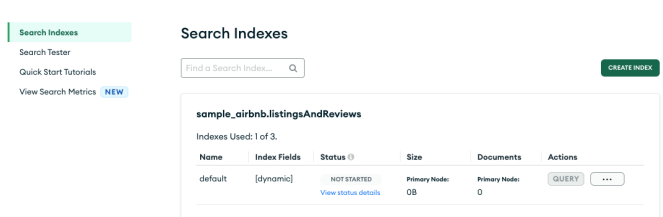
QUERY RESULTS: 1-20 OF MANY

```

{ "_id": "18086546"
  "listing_url": "https://www.airbnb.com/rooms/18086546"
  "name": "Ribeira Charming Duplex"
  "summary": "Fantastic duplex apartment with three bedrooms, located in the histor..."
  "space": "Privileged views of the Douro River and Ribeira square, our apartment ..."
  "description": "Ribeira duplex apartment with three bedrooms. Located in the histor..."
}

```

This enables *Atlas Search* for this data source, a fully managed Apache Lucene based fuzzy Search Index (in addition to Database Indexes that Atlas provides) - **Click all the Green default buttons as they appear (approx. 5) until Atlas tells you it is building your search index** . We are accepting the defaults for now.



Search Indexes

Search Tester Quick Start Tutorials View Search Metrics **NEW**

Find a Search Index...

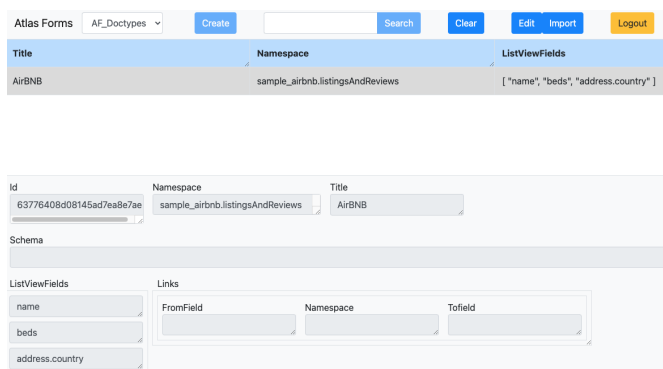
CREATE INDEX

sample_airbnb.listingsAndReviews

Indexes Used: 1 of 3.

Name	Index Fields	Status	Size	Documents	Actions
default	[dynamic]	NOT STARTED	0B	0	QUERY ...

Now return to your application and your custom application URL and add the sample_airbnb.listingsAndReviews data set to our application. **Choose AF_Doctype** from the dropdown at the top left then **Enter the following** in the form, **use the (+) to add a new box for each of the ListViewFields you enter. Leave Id blank.**



Atlas Forms AF_Doctype Create Search Clear Edit Import Logout

Title	Namespace	ListViewFields
AirBNB	sample_airbnb.listingsAndReviews	["name", "beds", "address.country"]

Id Namespace Title

63776408d08145ad7ea8e7ae sample_airbnb.listingsAndReviews AirBNB

Schema

ListViewFields Links

name FromField Namespace ToField

beds

address.country

Namespace:
sample_airbnb.listingsAndReviews

Title:

AirBNB

ListviewFields: (Three entries in list)

- **name**
- **beds**
- **address.country**

We are not going to enable relationships between document types just now but this is where to configure it..

Now **click Create** on the menu bar at the top. You just entered your first record in your system - this was a configuration record but data is entered the same way.

Refresh/ the page - this is the same as logging in again and you will now see your new form selected.

Enter a word like **"Ship"** in the text box next to the Search button and **click Search**. Now you see all properties that mention ships.

Click on one to view it. You can **click Clear** to try a different search, you can also search by filling in fields on a blank form.

As a Superuser you can create and edit anything, manage users, permissions, and dropdown lists. Let's configure a dropdown list - Change the dropdown to AF_Picklists and enter the following.

- Database: **sample_airbnb**
- Collection: **listingsAndReviews**
- Fieldname: **minimum_nights**
- Values: **1**
- 2**
- 3**
- 7**

Then **click Create** then reload the page. You will now see there is a dropdown for the Minimum nights field.

Finally as a Developer you can do far more - let's look at and then run a custom serverless function that generates picklists for us from the data in the collection. Open your App Services browser tab in the Atlas UI and click 'Functions' on the left to view your Atlas (Serverless) functions.

Click on the first one (ADMIN_PicklistGenerator), this function is private and is never called (or callable) from the UI although you could make it public and add a button. Around line 117 you can see it is hard coded to the collection we are working with - it will compute the values needed for all picklist fields. At the bottom of the Console, **Change User to yourself** then **Click Run** at the bottom of the page to run it and generate our picklists.

Now return to your Forms Application and **reload** and you will see many fields like *amenities* or *property type* have a list, and you can edit these in the AF_Picklists Doctype.

```

1- async function getPicklistValues(databaseName, collectionName, fieldName) {
2-   const collection = context.services.get("mongodb-atlas").db(databaseName).collection(collectionName);
3-   const SAMPLESIZE = 10000; /* Limit to 10,000 but not the first 10000 */
4-   let picklist;
5-   try {
6-     const sample = { $sample: { size: SAMPLESIZE } };
7-     const unwindArray = { $unwind: '$' + fieldName };
8-     const groupAll = { $group: { _id: null, count: { $sum: 1 }, values: { $addToSet: '$' + fieldName } } };
9-
10-    const pipeline = [sample, unwindArray, groupAll];
11-    picklist = await collection.aggregate(pipeline).toArray();
12-  } catch (e) {
13-    console.error(e);
14-  }
15-   return picklist[0];
16- }
17-
18- exports = async function (org) {
19-   //This is a private function, intended to be run from the App Services GUI only
20-   //Although you could use the code elsewhere - it's job is to generate a set of
21-   //Picklists - Version 1 we will give it an entity and fieldname and it
22-   //Will see how many unique values there are and if there are < 30 not counting
23-   //Blanks will generate a new Picklists
24-   const MAX_PICKLIST_LEN = 32;
25-   const database = "sample_airbnb";
26-   const collection = "listingsAndReviews";

```

Missing semicolon. System User Change User Run

```

> ran at 1668769775922
> took 821.43878ms
> logs:
186 unique elements from 121402
> result:
true

```

This is just scratching the surface of what you can do with Atlas Forms - The full manual is available at <https://mdb.link/AtlasFormsManual>