

# Qaas app deployment manual

This document is intended as a guide to deploy the Qaas app to the live environment.

## Initial Steps:

1. Switch the firebase options to the live environment in the flutter new client
2. Make sure there are no debug app check tokens in the index.html
3. Build the flutter client
4. Deploy the new flutter client to firebase, now the frontend is updated
5. Add / update / delete any firebase cloud functions for the new version into the live environment
6. Create API secrets in the live environment if needed
7. Allow secret access on the specific cloud functions
8. Make sure the firebase security rules are up to date in the live environment
9. Mirror cloud Firestore data in Algolia if needed.
10. Change allowed Algolia indexes in the Algolia API key functions, this decides which indexes youre allowed to search on with this specific Algolia search API key.
11. Build Firestore indexes if needed, for where / order by queries. This is usually done before client deployment

## Client code testing on the live environment

1. To test the client code on the live environment, create an App check debug token and place it in the index.html file. The path to the index file: web\index.html When going back to the test environment make sure to put the App check debug token back.
2. Proceed with doing step 2 and 3 from **Client code deployment to live environment**

## Building Firestore indexes

If you do not use indexes, you can skip this step.


- Building Firestore indexes is something I recommend doing when the client code is connected to the live environment. If you have a try catch block around your Firestore query, it will just give you the link to build the needed index.

- Firestore indexes only need to be build when you use orderBy within your query and some other scenarios. Indexes you can see as a copy of that collection where they sort the data beforehand. This is for performance reasons.

### Secret manager deployment (API keys)

1. Login to the Google Cloud console and make sure the right environment is selected **QaaS-Live\_omgeving-V2** <https://console.cloud.google.com/+>
2. Navigate to **Secret Manager** you can do this by typing secret manager in the search bar. At this page you see an overview of all the secrets that the selected environment has access to.
3. Click **Create Secret**
4. Give the secret a name and fill in the secret value. The secret value could be a password, API key. There are cases where usernames are also defined here. The reason for this is that usernames then can be changed without redeploying the whole code base.
5. Make sure to check **Manually manage locations for this secret** and then select **Europe-west3** be sure to always deploy resources within Europe.
6. When you are done click **Create secret** in the bottom of the page to save the secret.

### Adding a new secret version (API keys)

1. Follow step 1 to 2 from **Secret manager deployment (API keys)**
2. On the Secret manager page click the action button on the secret you desire to add a new version on. This is an icon with triple dots 
3. Then click on **Add new version**, this will open a dialog where you can fill in the new secret value.
4. Make sure you Disable all past versions of the secrets.
5. Click **Add new version** to save the new version

## Cloud function code deployment

1. Make sure you are in the root of the Cloud function directory
2. Login to Firebase through the visual studio code terminal. First make sure you are in the root project folder. Then type **Firebase login**, this will either make sure you are logged in to the right account already or you can log in to the correct Firebase project. If you are already logged into a different project, type **Firebase logout** to logout and then login to the correct project.
3. Make sure you are connected to the correct project through the terminal by using the **Firebase use** command. This will list the current active project. If you want to switch use **Firebase use <project id>** for example **Firebase use qaas-live-omgeving-v2**.
4. Use **firebase deploy --only functions:<functionName>** to deploy a specific function. **Firebase deploy --only functions** would deploy redeploy all the functions, but I would not recommend this. Only do this if necessary and you have a good understanding of what all the functions do.
5. Open the Firebase project in the browser  
<https://console.firebase.google.com/project/qaas-live-omgeving-v2/functions> and navigate to the Cloud functions tab. If deployed correctly the function is visible here.
6. Hover over the function and on the right should be an action button visible. Here you can view the **logs** or view the **detailed usage stats**. Any console.log will appear in the logs of the function.
7. In case you have **Secrets** that the cloud function is using from the secret manager, you need to assign these secrets to the cloud function. You do this by going to the **detailed usage stats** of the cloud function. Then on the top you click **Edit**. You will see **Runtime, build, connections and security settings** click this open. Now you see multiple tabs. Click on **Security And Image Repo**. Here you can add **Secrets** to the cloud function. If you don't do this, you will get permission errors shown in the console logs. Make sure to click next and then click on deploy again, this deploys the functions with access to the assigned secrets. I wonder if there is a better way of doing this in one go. But it does add a security layer where you must do a specific action to grant access.

## Cloud Firestore rules deployment

1. Open the firebase web console <https://console.firebase.google.com/>
2. Make sure you selected the correct environment **QaaS-Live-omgeving-V2**
3. In the Firebase project navigate to **Firestore Database**
4. Click on **Rules** you can find this in the tabs on the top.
5. Here you see all the security rules that are applied on the **Cloud Firestore** of the selected environment. These rules determine who can Read, Write, Update, Delete data from the cloud Firestore. This can be defined for each specific collection.
6. If you want to make change you will probably copy the rules from the test environment into this after a good review. So, you tested your change already in the test environment.
7. When you hit ctrl + s you save the rules, and they will be deployed to the live environment. Beware that it can take several minutes before cloud Firestore rules take effect. The previous rules are still there in the version control on the left of the screen in case we made a mistake, and we need to revert the made changes.

## Cloud storage rules deployment

The cloud storage security rules are quite like the Firestore security rules. Most steps are the same.

1. Open the firebase web console <https://console.firebase.google.com/>
2. Make sure you selected the correct environment **QaaS-Live-omgeving-V2**
3. In the Firebase project navigate to **Storage**
4. Click on **Rules** you can find this in the tabs on the top.
5. Here you see all the security rules that are applied on the **Storage** of the selected environment. These rules determine who can Read, Write, Update, Delete data from the cloud storage. This can be defined for each specific path or document.
6. If you want to make change you will probably copy the rules from the test environment into this after a good review. So, you tested your change already in the test environment.
7. When you hit ctrl + s you save the rules, and they will be deployed to the live environment. Beware that it can take several minutes before cloud Firestore rules.
8. The Cloud storage makes use of custom claims mostly for access. But when deploying these rules, you already should be aware of what these are. If this is not known, here is some information

<https://firebase.google.com/docs/auth/admin/custom-claims> custom claims are defined for each user when their account is created or updated. This is done in a cloud function.

## Extension deployment (Algolia)

1. Open the firebase web console <https://console.firebase.google.com/>
2. Make sure you selected the correct environment **QaaS-Live-omgeving-V2**
3. In the Firebase project navigate to **Extensions**
4. Click Explore extensions
5. Search for **Search Firestore with Algolia** in this example and click on install
6. Select the correct environment to install the extension on **QaaS-Live-omgeving-V2**
7. Name the extension
8. Fill in the collection path. So, for example you have a collection named **User**
9. You could specify the indexable fields but usually we keep this empty and index all
10. Name the Algolia index name equal to the collection name in Firebase, so in this example it would be **User**.
11. Retrieve the Algolia Application Id
  - a. Go to <https://www.algolia.com/> and sign in
  - b. Make sure you select the correct environment **Qaas Live omgeving**
  - c. Click on settings on the bottom left of the page
  - d. Click API keys
  - e. On the top of the screen the Application ID is visible, copy and paste this into the Firebase extension page.
12. Retrieve the Algolia API key “We recommend creating a new API key with “addObject”, “deleteObject”, “listIndexes”, “deleteIndex”, “editSettings”, and “settings” permissions. **Do not use the Admin API key.**
  - a. You can find this in the same place as the Application ID. But instead navigate to All API keys
  - b. Here is an API key with the name **Firebase live omgeving api key**, use this key
13. Optionally you can select Yes on Full index existing documents. But this is only if you already have data in the firestore collection you want to install Algolia on.
14. Make sure you select **Europe-west3** as Cloud function location
15. Click **Install extension** to start installing the extension, this will take several minutes.

## Algolia commonly used settings

1. Make sure you have selected the correct Algolia environment **Qaas live omgeving**

2. On the overview page scroll down to the indexes. Only indexes with data will be shown here. So, if the Firestore collection is empty. So will be the index here. Algolia mirrors the Firestore collection.
3. Click on your desired index / collection
4. Click on Configuration
5. Add searchable attributes, so if you want to search on first name only. You add first name here as searchable attribute.
6. Tip: is searching not working in the QaaS app? Go to the browse tab within Algolia on the desired index / collection and use the search bar there for testing. If the result is found there. You know your Algolia settings are at least setup correctly.

## Client code deployment to preview channels

Preview channels are used for testing new live versions. Preview channels get their own separate links. They also have an expire time which can be set in the command. When deploying a preview channel, the existing live version will not change.

1. First make sure you do step 1 to 6 from **Client code deployment to live environment**
2. To deploy the client code to a preview channel make use of the following command  
**firebase hosting:channel:deploy QaaSv0.8 --expires 7d** “QaaSv0.8” is the name of the preview channel, use this for tracking the channel. “—expires 7d” means it will expire in 7d and it will automatically be deleted after that.
3. You will now receive a link where you can access the newly deployed version on the preview channel.
4. To make authentication work on the preview channel you need to add the url to list of allowed urls that can make use of the public API key:  
<https://console.cloud.google.com/apis/credentials?project=qaas-live-omgeving-v2> you need to change this on the **Browser key (auto created by Firebase)** make sure to save your changes
5. You also need to add the url to Google reCAPTCHA  
<https://www.google.com/recaptcha/admin/site/483217617/settings> without this the MFA would not be working correctly. It can take some time before these changes take effect. Wait a couple of minutes. Also make sure to do ctrl + f5 to be sure it's not a cache problem.
6. After these steps you should be able to login to the Qaas app on the preview channel
7. If everything is working correctly and you want to deploy the preview channel to the current live channel, all you have to do is **firebase hosting:clone qaas-live-omgeving-v2:QaaSv0.8 qaas-live-omgeving-v2:live**

## Client code deployment to live environment

Note: this is **ONLY** client code deployment. This does **NOT** include Cloud functions code, Secrets from the secret manager, Cloud firestore rules, any extensions, Algolia settings, Building firestore indexes, Cloud storage rules, etc.



1. Make sure there is no `FIREBASE_APPCHECK_DEBUG_TOKEN` in the `index.html` this looks something like this: `<script>self.FIREBASE_APPCHECK_DEBUG_TOKEN = "the token is here";</script>`. Remove the whole line including the script tags, be sure to save the document. The path to the index file: `web\index.html`
2. Check which environment the client is connected to. You can do this by looking at the `firebase_options.dart` file located at: `lib\firebase_options.dart`. **ProjectId: 'afstudeerapp-6df3b'**, means its connected to the test environment. The live environment is **ProjectId: 'qaas-live-omgeving-v2'**
3. If you want to switch environments in the `firebase_options` file does one of the following commands **`flutterfire configure -p afstudeerapp-6df3b`** or **`flutterfire configure -p qaas-live-omgeving-v2`**
4. Login to Firebase through the visual studio code terminal. First make sure you are in the root project folder. Then type **Firestore login**, this will either make sure you are logged in to the right account already or you can log in to the correct Firebase project. If you are already logged into a different project, type **Firestore logout** to logout and then login to the correct project.
5. Make sure you are connected to the correct project through the terminal by using the **Firestore use** command. This will list the current active project. If you want to switch use **Firestore use <project id>** for example **Firestore use qaas-live-omgeving-v2**.
6. Run **`flutter build web`** to prepare the client code for deployment.
7. **This next move will deploy the client code to the live environment** which is used actively by users. Run **Firestore deploy** to deploy to the live environment. The **Firestore use** command from before made sure we are deploying this to the correct environment.