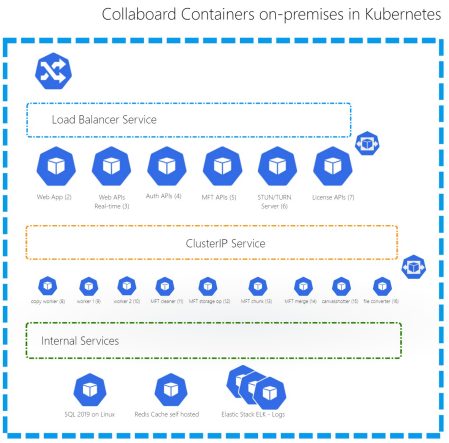
Collaboard
Great minds think together

Title	Install guide Collaboard
Abstract	Describes how to install and use Collaboard on the different containerized offerings
Status	Draft
Document Owner	IBV
Document Author(s)	Noel Hermans, Dimosthenis Stellakis, Václav Šedivec, Dennis Vroegop
Document Approver	G.P. Santopaolo

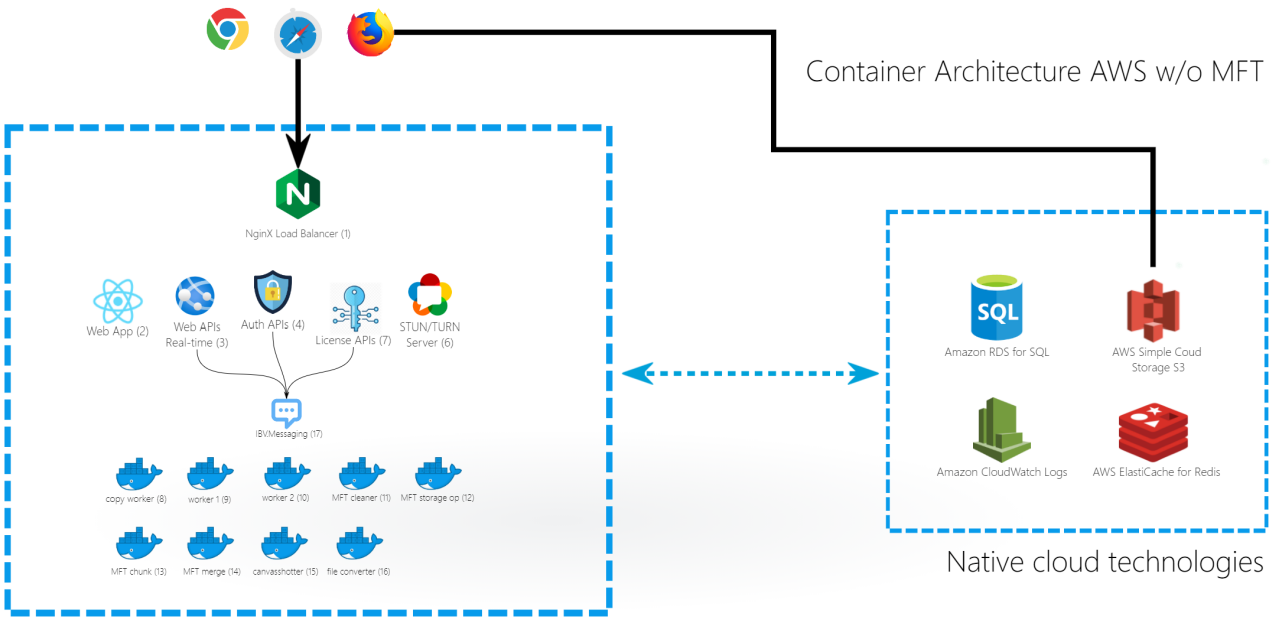
Collaboard Application

Collaboard, from an architectural point of view, is an extremely flexible application. It can run on any cloud using containers by taking advantage of native cloud technologies. AWS ECS Fargate, AWS EKS, Azure AKS, Container Instances, Red Hat OpenShift, Google GKE, any Kubernetes distribution running on-premise, or even Docker-compose (for very small POCs) are all supported scenarios where Collaboard can be installed. This document will guide you through the installation process of the scenario you selected as appropriate for your needs. An example of Collaboard's flexibility is the use of storage. We all know that each cloud provider has its offering when it comes to storage. With Amazon S3 or Azure Blob Storage, it's effortless to transfer terabytes of data across clients and servers all over the World. When it comes to on-premises, it is impossible to achieve the same results. For this specific use case, we developed our custom native technology so that Collaboard can handle file storage on-premises with the same simplicity offered by the cloud.

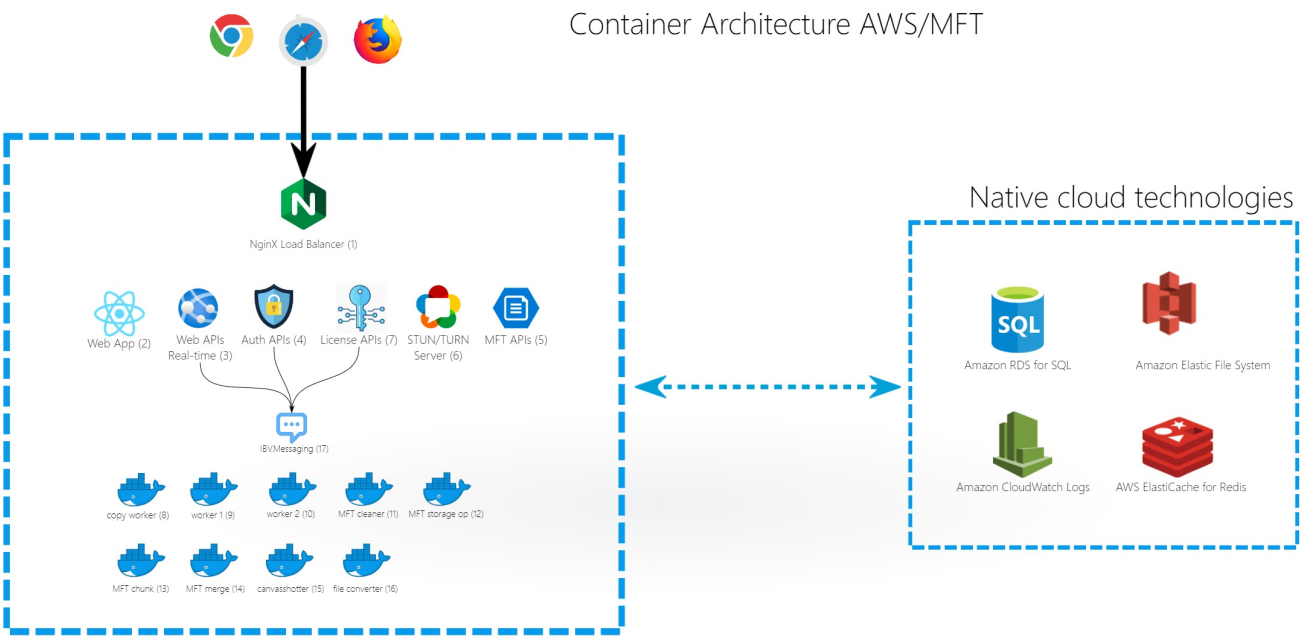


The diagram below shows Collaboard running on a full on-premises Kubernetes/OpenShift cluster

The diagram below shows Collaboard running on AWS using some native cloud technologies, including the S3 storage



The diagram below shows Collaboard running on AWS without exposing the S3 storage to the clients. For the use case below, the clients use IBV MFT (Managed File Transfer) native technology to perform all the upload and download operations.



As for storage, we offer a wild range of options for several components of the application, such as the authentication providers or the messaging services. We can also run our application on-premises self-hosted on a variety of different containerized solutions. This document helps you plan your installation, choose and configure the right set of components that best fits your requirements, scalability, and reliability needs.

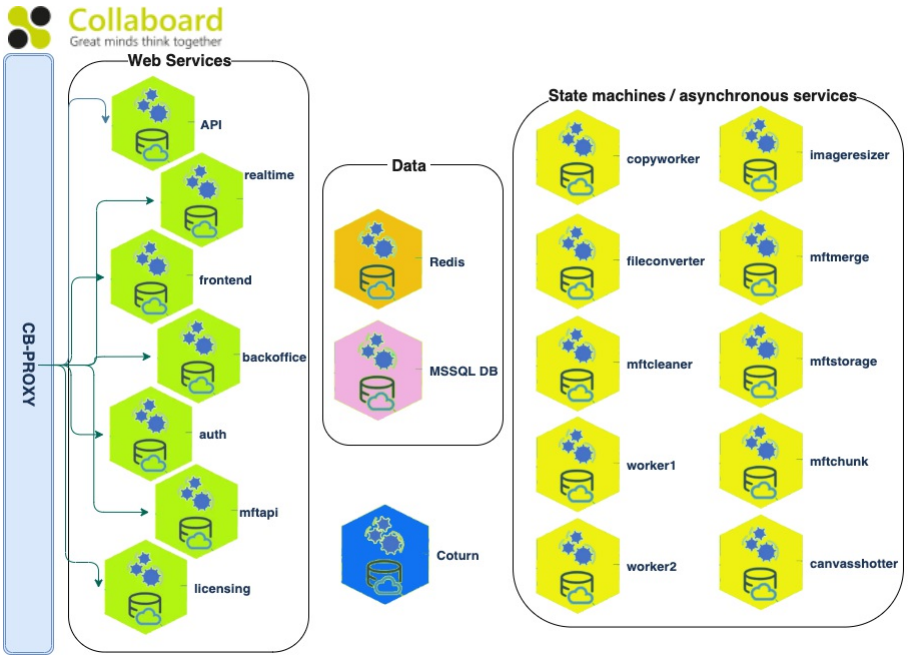
Components

Collaboard counts several containers that can be grouped into four categories:

- Frontend servers
- Web endpoints
- Data
- State machines / asynchronous services

Design

The diagram below is a visual representation of Collaboard's architecture



The following table list the containers used by Collaboard. Note: the containers may vary depending on the installation requirements/configuration and on the application version.

Name	Type	Remarks
Nginx	Proxy server	Handles all traffic from the outside world to the internal containers running on the Docker network "CBNetwork". This is the only open part of the system.
API	HTTP Server	Main API for the system. Reached from the outside through the proxy via https:///server/api
API/SignalR	HTTP Server	The realtime component. Reached from the outside through the proxy via https:///server/signalr
Licensing	HTTP Server	Takes care of the licensing handling. Reached from the outside through the proxy via https:///server/licensing
Auth	HTTP Server	Handles all authentication. Reached from the outside through the proxy via https:///server/auth
MFTApi	HTTP Server	Handles all file and bloc storage and retrieval. Reached from the outside through the proxy via https:///mft
Frontend	HTTP Server	The main frontend server, serving the React files. Reached from the outside through the proxy via https:///
Backoffice	HTTP Server	The backoffice server, serving the admin console. Reached from the outside through the proxy via https://admin./
DB	Database	Database server. Not accessible from the outside. Internal open port 1443. Running SQL Server 2017
Copy	Background service	Handles the asynchronous copying of tiles and projects.
Image Resizer	Background service	Handles the resizing of the images for a responsive system.
File Converter	Background service	Handles the asynchronous conversion of Office files to images for thumbnail display.
MFT Merge	Background service	Handles the merging of all bits retrieved for the blob storage
MFT Storage Service	Background service	Handles the asynchronous tasks for handling all blob storage
MFT Cleaner	Background service	Handles the asynchronous tasks for cleaning up the residual files for the blob storage
Worker 1	Background service	Assorted background processes
Worker 2	Background service	Assorted background processes
Canvas Shotter	Background service	Handles the asynchronous generation of thumbnails for the projects
Coturn	HTTP server	STUN/TURN server for WebRTC. Reached from the outside on ports 3478 (HTTP) and 5349 (HTTPS). Additionally uses a range of ports for WebRTC connections
Redis	HTTP Server	Accessible on the closed network only, used by API and API/SignalR containers

Since we are working to improve the application and to add new features the number of containers described in this document may vary at the moment you will install the application.

[[*TOC*]]

Scope

When installing for the first time Collaboard it is important to fulfill the requirements list and have all the requirements prepared so that they can be used during the installation process. This page provides a list of all the requirements. At the bottom of each requirement is a reference to the configuration ("**Configured in**"). This will make it easier to understand where each requirement is configured.

DNS Names

Prepare DNS name for accessing the Collaboard application The applications need minimum of **two DNS names**: One for each web application. example: https://web.collaboard.app (**frontend**) and https://admin.collaboard.app (**backoffice**)

In following sections and other parts of documentation, APIs are assumed to be served from **/server** virtual path. example:

- https://web.collaboard.app/server for main API
- https://web.collaboard.app/server/auth for authentication API
- ...

Optionally the APIs can be served from separate domain(s), but that needs to be reflected in all steps of overall configuration. **Configured in** : *env.Override.js ; appsettings.Override.json ; ConfigurationMigration.sql*

SSL certificates

Create a SSL certificate for accessing each of the Collaboard URLs (DNS Names). Important note: by design the application supports only trusted CA signed certificates. We do not support self-signed certificates in any form and for any usage. There are options to buy trusted CA signed certificates with a low budget or even for free.

Create an SSL certificate for each of the Collaboard DNS names mentioned above. As a result, you shall provide two files:

- The private key (e.g. cb_proxy.key)
- The SSL certificate (cb_proxy.crt)

Depending on the authentication provider you are opting to use you may have a need for more certificates. You can find more about the matter in the next chapter, the Authentication paragraph. **Configured in** : *envoy.Override.yaml* In case you want to install a TURN server, you need additional SSL certificated as well. This has been explained in the TURN server section.

Authentication Providers

Collaboard supports integration with Google, Microsoft, and Apple. We also support any third-party identity providers that support OAuth+OpenID or SAML 2.0 protocols. The configuration for the external providers for the server API is set using the `Server_WebApiExternalLoginProviders` configuration setting in the `dbo.t_configuration` table. This setting is a JSON array serialized as a string, with the following structure:

```
[
   {
      "AppCode":"webapp",
      "LoginProviders":[
         {
            ...
         },
         {
            ...
         }
      ]
   },
   {
      "AppCode":"webapp-admin",
      "LoginProviders":[
         {
            ...
         },
         {
            ...
         }
      ]
   }
]
```

In the examples below, we are assuming that

- Collaboard web application is hosted on `https://collaboard-app.customer.com/`
- BackOffice web application is hosted on `https://collaboard-admin.customer.com/`
- Collaboard authentication API is hosted on `https://collaboard-app.customer.com/server/auth`

In case of a self-hosted ADFS or SAML provider, the identity server will need to support HTTPS communication with an SSL certificate from a trusted certification authority.

Following steps are prepared for **webapp**. For **webapp-admin** follow the exact same steps with one simple modification: replace *collaboard-app.customer.com* occurrences with *collaboard-admin.customer.com*

Google

For Google sign-in, the customer needs to create a web app in the Google app console. The steps are:

1. Go to the Google API Console
2. Open the Credentials page.
3. Click Create credentials > OAuth client ID.
4. Select the Web application type.
5. Name your OAuth 2.0 client application.
6. Add the Authorized redirect URI (e.g. https://collaboard-app.customer.com/authenticate)
7. Click Create
8. Copy the Client ID and Client secret displayed

The provider object shall have the following format:

```
{
    "Provider": "Google",
    "ClientId": <The Google client ID>,
    "ClientSecret": <The Google client secret>,
    "RedirectUri": "https://collaboard-app.customer.com/authenticate",
    "AuthorizeUri": "https://accounts.google.com/o/oauth2/v2/auth",
    "TokenUri": "https://www.googleapis.com/oauth2/v4/token",
    "Scope": "openid email profile",
    "TFAEnabled": <true if we want extra two-factor authentication after logging in with ADFS, false otherwise>
}
```

In the above example, you need to replace the `collaboard-app.customer.com` with the domain in which you are hosting the Collaboard web application.

Microsoft

For Microsoft sign-in, the customer first needs to create a web app in Microsoft active directory. The steps are:

1. Go to the Azure portal
2. Select Azure Active Directory.
3. Select App Registrations and click on New Registration.
4. Name your OAuth 2.0 client application.
5. Select the appropriate accounts type.
6. Add the Authorized redirect URI (e.g. https://collaboard-app.customer.com/authenticate)
7. Click Create
8. Copy the Application (client) ID
9. Click "Certificates and secrets"
10. Click "New client secret", select the expiration date and copy the Value displayed.

The provider object shall have the following format:

```
{
    "Provider": "Microsoft",
    "ClientId": <The Microsoft client ID>,
    "ClientSecret": <The Microsoft client secret>,
    "RedirectUri": "https://collaboard-app.customer.com/authenticate",
    "AuthorizeUri": "https://login.microsoftonline.com/common/oauth2/v2.0/authorize",
    "TokenUri": "https://login.microsoftonline.com/common/oauth2/v2.0/token",
    "Scope": "openid email profile",
```

```
    "TFAEnabled": <true if we want extra two-factor authentication after logging in with ADFS, false otherwise>
}
```

In the above example, you need to replace the `collaboard-app.customer.com` with the domain in which you are hosting the Collaboard web application.

Apple

For Apple sign-in, the customer first needs to create an Apple app registration. For instructions on how to do this, please see https://medium.com/identity-beyond-borders/how-to-configure-sign-in-with-apple-77c61e336003

The provider object shall have the following format:

```
{
    "Provider": "Apple",
    "ClientId": <The Apple client ID>,
    "ClientSecret": <The Apple client secret>,
    "RedirectUri": "https://collaboard-app.customer.com/server/auth/api/Authorization/HandleExternalLoginCallback?app=webapp&provider=Apple&redirectUri=https%3A%2F%2Fcollaboard-ap
    "AuthorizeUri": "https://appleid.apple.com/auth/authorize",
    "TokenUri": "https://appleid.apple.com/auth/token",
    "Scope": "name email",
    "TFAEnabled": <true if we want extra two-factor authentication after logging in with ADFS, false otherwise>
}
```

Because Apple is using POST to send back the response, the web api needs to handle the response before sending it back to the client, so the redirect URI will be in the form:

`{Collaboard authentication API URI}`/api/Authorization/HandleExternalLoginCallback?app=webapp&provider=Apple&redirectUri=`{URL-encoded value of the Collaboard web application root URI}`%2Fauthenticate

ADFS

Important: Self-signed certificates are not supported.

For ADFS, the provider object shall have the following format:

```
{
    "Provider": "ADFS",
    "ClientId": <The ADFS application id for Collaboard>,
    "ClientSecret": <The ADFS secret key for the Collaboard app>,
    "RedirectUri": "https://collaboard-app.customer.com/authenticate",
    "AuthorizeUri": <The ADFS authorize URI>,
    "TokenUri": <The ADFS token URI>,
    "Scope": "openid email profile",
    "TFAEnabled": <true if we want extra two-factor authentication after logging in with ADFS, false otherwise>
}
```

In the above example, you need to replace the `collaboard-app.customer.com` with the domain in which you are hosting the Collaboard web application.

Also, in the **id_token** returned by ADFS, the claims needed are:

Claim	Description	Required
unique_name *or* preferred_username	The user name	Yes
email	The user's email	Yes
given_name	The user's first name	No
family_name	The user's last name	No
name	The user's full name	No
picture	The user's profile photo url	No

SAML

To integrate with an identity provider using the SAML 2.0 protocol, we will need the XML Metadata of the provider. A sample IdP XML metadata file would look like this:

```
<EntityDescriptor entityID="urn:idp.example.org" xmlns="urn:oasis:names:tc:SAML:2.0:metadata">
  <IDPSSODescriptor protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
    <KeyDescriptor use="signing">
      <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
        <X509Data>
          <X509Certificate>MIIDBzCCAe+gAwIBAgI...P3Z3TTTs=</X509Certificate>
        </X509Data>
      </KeyInfo>
    </KeyDescriptor>
    <SingleLogoutService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect" Location="https://idp.example.org/saml/logout"/>
    <SingleLogoutService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST" Location="https://idp.example.org/saml/logout"/>
    <NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress</NameIDFormat>
    <NameIDFormat>urn:oasis:names:tc:SAML:2.0:nameid-format:persistent</NameIDFormat>
    <NameIDFormat>urn:oasis:names:tc:SAML:2.0:nameid-format:transient</NameIDFormat>
    <SingleSignOnService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect" Location="https://idp.example.org/saml"/>
    <SingleSignOnService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST" Location="https://idp.example.org/saml"/>
    <Attribute Name="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri" FriendlyName="E-Mail Address'
    <Attribute Name="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri" FriendlyName="Given Name" xmlns=
    <Attribute Name="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri" FriendlyName="Name" xmlns="urn:oasis
    <Attribute Name="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri" FriendlyName="Surname" xmlns="urn
    <Attribute Name="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri" FriendlyName="Name ID" xmlns
  </IDPSSODescriptor>
</EntityDescriptor>
```

Alternatively, we would need the following information from the identity provider:

- IdP EntityID
- IdP Redirect URL
- IdP Logout URL
- IdP public certificate
- NameId Format
- User-specific attributes returned by the authentication response

The provider JSON object will be:

```
{
    "Provider": "SAML",
    "ClientId": <The EntityID for the Collaboard application>,
    "ClientSecret": <The IdP public X509 certificate value>,
    "RedirectUri": "https://collaboard-app.customer.com/server/auth/api/Authorization/HandleExternalLoginCallback?app=webapp&provider=SAML&redirectUri=https%3A%2F%2Fcollaboard-app
    "AuthorizeUri": <The IdP login URI>,
    "TokenUri": null,
    "Scope": null,
    "TFAEnabled": <true if we want extra two-factor authentication after logging in with SAML, false otherwise>
}
```

In the SAML case, because the IdP is using POST to send back the response, the web API needs to handle the response before sending it back to the client, so the redirect URI will be in the form:

`{Collaboard auth API URI}`/api/Authorization/HandleExternalLoginCallback?app=webapp&provider=SAML&redirectUri=`{URL-encoded value of the Collaboard web application root URI}`%2Fauthenticate

For the client API, the customer needs to edit the env.js file and set the following properties:

Collaboard app: `apiUrl:`, e.g. "https://collaboard-app.customer.com/server", `appName`: "webapp", `authProviders:`, e.g. ["ADFS", "SAML"]>

BackOffice app: `apiUrl:`, e.g. "https://collaboard-app.customer.com/server", `appName`: "webapp-admin", `authProviders:`, e.g. ["ADFS", "SAML"]>

Configured in : *ConfigurationMigration.sql ; env.Override.js*

Security

Decide how much time the authentication token and the one-time password shall last. One time password duration is configurable via **Server_WebApiOTPDurationMins** value in *dbo.t_configuration* table. It sets the duration for reset password code validity. Values for **Server_WebApiAuthTokenDurationMins** and **Server_WebApiRefrTokenDurationMins** configure respective durations. In short, access token decides how long after login the token is valid. After it expires, it needs to be refreshed using the refresh token. Refresh token duration sets the cap on how long user can return to his current session before being logged out. **Configured in** : *ConfigurationMigration.sql*

Licensing

Based on your agreement you shall have a licensing file provided by IBV. The DevOps team shall provide a license file that you can use to replace the default license file that came with the installation. This will enable the Collaboard features for your installation based on the license you have purchased. The file needs to be mounted to a container folder of your choosing (we suggest */var/overrides* since it's also already required on most of containers) and it's path (i.e. */var/overrides/license.lic*) needs to be specified in **Server_LicensingLicenseProvider** value in *dbo.t_configuration* table. **Configured in** : *ConfigurationMigration.sql*

User to be Admin

In order to activate license, the team owner has to be selected and used during license initialization. Optionally one or more license managers should be selected as well. These will be able to access and manage the application using BackOffice.

Team members can be added to the license in two ways.

- Team owner invites the users from team management tab of Collaboard app.
- License manager can use backoffice to assign users to license directly.

Configured in : *ConfigurationMigration.sql*

Email

Collaboard sends emails to users for various reasons: including registration, password recovery, and various notification. To be able to send emails the customer shall provide one supported email provider:

- SMTP
- SendGrid
- SendInBlue

The customer shall provide the right credentials/API Keys once the provider is decided.

Providers

Collaboard currently supports sending emails using SMTP, SendGrid and SendInBlue. The configuration setting for this is `Server_MailProviderConnectionString` and each provider's configuration is described below:

Provider	Server_MailProviderConnectionString setting value
SMTP	Provider=SmtpMailProvider;Host=`<SMTP host>`;Port=`<SMTP port number>`;EnableSsl=`<true | false>`;Username=`<SMTP username>`;Password=`<SMTP user password>`
SendGrid	Provider=SendGridMailProvider;ApiUrl=https://api.sendgrid.com;ApiKey=`<SendGrid API key>`
SendInBlue	Provider=SendInBlueMailProvider;ApiUrl=https://api.sendinblue.com/v3;ApiKey=`<SendInBlue API key>`

Templates

The customer can customize the messages sent by Collaboard, using the `Server_MailTemplates` configuration setting. The value for this setting is a JSON array serialized as a string, with the following layout:

```
[{
    "Theme": "default",
    "Templates": [
        { "Code": "ResetUserPassword", "Language": "en-US", "Sender": "info@customer.com", "Recipient": "{Recipient}", "Subject": "", "Body": "", "ExternalID": "" },
        ...
    ]
}, {
    "Theme": "theme1",
    "Templates": [
        { "Code": "ResetUserPassword", "Language": "en-US", "Sender": "info@customer.com", "Recipient": "{Recipient}", "Subject": "", "Body": "", "ExternalID": "" },
        ...
    ]
}]
```

The template JSON object structure is the following:

Property	Description	Example value
Code	The template's code	ResetUserPassword
Language	The template's language	en-US
Sender	The sender	info@customer.com
Recipient	The sender	
Subject	The email subject	We got a request to reset your Collaboard password
Body	The email body	You can use the code to reset your password. This code will be valid for minutes
ExternalID	The ID of the template in the external message provider	abcdefg

Collaboard supports both internally defined and externally defined message templates. An internal message template means that the template subject and body is defined internally in the Collaboard configuration. This could be used when using a mail provider that does not have support for creating message templates, for example SMTP. An external message template means that the template is defined in the message provider and that Collaboard references it.

Example of an internal message template:

```
{
   "Code":"ResetUserPassword",
   "Language":"en-US",
   "Sender":"info@customer.com",
   "Recipient":"{Recipient}",
   "Subject":"We got a request to reset your Collaboard password",
   "Body":"You can use the code {Token} to reset your password. This code will be valid for {TokenDuration} minutes."
}
```

For an internal message template, the ExternalID property needs to be unassigned/null and all the other properties to have appropriate values. In the values, you can set a variable inside curly brackets (e.g. `{XXXX}`) and it will be replaced during the email sending with its corresponding value if any.

Example of an external message template:

```
{
   "Code":"ResetUserPassword",
   "Language":"en-US",
   "Sender":"info@customer.com",
   "Recipient":"{Recipient}",
   "ExternalID":"abcdefg"
}
```

In the `ExternalID` property, you should put the ID of the message template defined in the provider of your choice. The properties Subject and Body are not required, provided that they are set in the message template defined in the provider.

Available message templates and their variables

ResetUserPassword: Sent when there is a request from a user to reset his password. The code passed the following variables to the template:

Variable	Description
Recipient	The recipient's email address
FirstName	The recipient's first name
LastName	The recipient's last name
Language	The template's language
Token	The reset password token
TokenDuration	The token's duration
RedirectUrl	The Collaboard web application's landing URL for resetting the password

VerifyUserAccount: Sent when a user registers, or when there is a need to verify the user's account. The code passed the following variables to the template:

Variable	Description
Recipient	The recipient's email address
FirstName	The recipient's first name
LastName	The recipient's last name
Language	The template's language
Token	The verification token
TokenDuration	The token's duration
RedirectUrl	The Collaboard web application's landing URL for verifying the user's account

SendUserOTP: Sent when a user requests a new OTP token to login using two-factor authentication. The code passed the following variables to the template:

Variable	Description
Recipient	The recipient's email address
FirstName	The recipient's first name
LastName	The recipient's last name
Language	The template's language
Token	The OTP token
TokenDuration	The token's duration

InviteProjectParticipant: Sent when a user is invited to a project via email. The code passed the following variables to the template:

Variable	Description
Recipient	The recipient's email address
InviterEmail	The email address of the person that created the invitation
ProjectName	The name of the project
Permission	"read" or "write"
InvitationUrl	The Collaboard web application's landing URL for accepting the invitation
Notes	Any notes entered by the user that created the invitation

Licensing.AddSubscriptionUser: Sent when a new user is added to a subscription. The code passed the following variables to the template:

Variable	Description
Recipient	The recipient's email address
FirstName	The recipient's first name
LastName	The recipient's last name
Language	The template's language
PlanName	The name of the subscription plan
PlanDuration	The duration of the subscription plan
SubscriptionOwnerFullName	The full name of the subscription owner
InvitationAcceptUrl	The Collaboard web application's landing url for accepting the invitation
InvitationRejectUrl	The Collaboard web application's landing url for rejecting the invitation

Licensing.NewPurchase: Sent when a user completes a new plan purchase. The code passed the following variables to the template:

Variable	Description
Recipient	The recipient's email address
FirstName	The recipient's first name
LastName	The recipient's last name
Language	The template's language
LicenceName	The name of the subscription plan

Configured in : *ConfigurationMigration.sql*

Storage

Collaboard can be deployed both on-premises or on the cloud. The customer shall provide the appropriate storage location to host both the database files and the files stored into the application.

Storage for files

In case Collaboard will be installed on **Kubernetes or OpenShift**, it is necessary to provision persistent volume claims (PVC) to the application with the following specifications :

- Minimum 2x 5 GB **ReadWriteMany** volumes (e.g. NFS, EFS, ... storage) in order to store the files. This **pvc** needs to be shared across several containers.

Storage for the MSSQL database

This will be only necessary in case you want to run the database in **Kubernetes or OpenShift** :

- Minimum 2x 10 GB **ReadWriteOnce** volumes (e.g. CSI, EBS, ... storage) for the Database and log volumes. This pvc can only be mounted by the **database** container.

Configured in : Kubernetes or OpenShift

Database

Microsoft SQL Server 2019 License

The minimum requirement for production installations above 250 users is SQL Standard + 10 call. For a POC installation or for a production installation for less than 250 users it is possible to use the free version of SQL Server 2019 (Express) There are two ways the customer can decide to host the Colllaboard database:

1. Hosted into the Collaboard cluster in a container. In this case, we will provide a container with a fully working SQL Server 2019 where the sure shall accept the EULA based on the purchased license (SQL License costs are not included in the Collaboard offer).
2. Hosted into a pre-existing SQL Engine provided by the customer. In this case, there is no need for a license. It's the customer's duty to provide a DB engine with the necessary resources to fulfil the application requests.

SQL Server location

There are two options regarding the DB: Option 1 : Have the database hosted externally, which might be preferred in most cases due to manageability, backups, failover etc. Option 2: Have the database running as a dedicated container.

- External: prepare the connection string of your external DB, fixed port is required example : Server=**database1.server.com**,**1433**;Initial Catalog=**IBV.Database**;Persist Security Info=False;User ID=**sa**;Password=**XXXXXXXXX**;Integrated Security=false;Connection Timeout=30;
- Container: make sure you have a license for it example : Server=**db**,**1433**;Initial Catalog=**IBV.Database**;Persist Security Info=False;User ID=**sa**;Password=**XXXXXXXXX**;Integrated Security=false;Connection Timeout=30;

Note: all DB connection parameters are configurable : Server, Port, Instance, User, Password.

SQL Server EULA and version

If you decide to run the SQL Server with Collaboard DB container, there are three parameters in the corresponding orchestrator manual (Docker-Compose, Kubernetes, or Openshift) that you need to set properly:

- ACCEPT_EULA for accepting the SQL Server's terms and condition
- MSSQL_PID to set licensing of your choosing/purchase, allowed values here
- SA_USERNAME to specify the database username. If not specified, it will use "sa" by default.
- SA_PASSWORD to specify the database password. This applies to both cb_db and cb_initialize_db containers

Configured in : *appsettings.Override.json*

Redis Cache

If you are going to deploy the application with scalability requirements (more than 200 users) then we need to activate the Redis cache. Scenarios where Redis is required:

- Deployment of both the Web API Container and the real-time container are both deployed.
- When the Web API Container and the real-time container are configured to scale up. Scenarios where the Redis cache is not required:
- Deployment of both the Web API Container only and the real-time container is not used.
- When the Web API Container is not configured to scale up and there is no use of the real-time container. Use cases where the Redis Cache is not used are typically when the application is deployed on Docker-Compose for a POC or in production for less than 250 users.

Configured in : *ConfigurationMigration.sql*

Azure CLI

Depending on the OS the proper Azure CLI shall be installed

- Install the Azure CLI for Windows | Microsoft Docs
- Install the Azure CLI for macOS | Microsoft Docs

Orchestrator CLI utilities

- In case for Docker-compose : https://docs.docker.com/compose/install/ **Important note:** It is important to install a recent version of docker-compose, otherwise the deployment will fail. We are using version 3 of the docker-compose.yml file which corresponds with Docker 19.03.0+ and docker-compose 1.28+
- In case for Kubernetes : https://kubernetes.io/docs/tasks/tools/install-kubectl/
- In case for OpenShift : https://docs.openshift.com/container-platform/latest/cli_reference/openshift_cli/getting-started-cli.html

Docker Desktop

Only for local/desktop installation via Docker-compose

- Docker Desktop for Windows - Docker Hub
- Docker Desktop for Mac - Docker Hub

Access to the IBV containers registry

Contact DevOps/Azure team to request access to your account.

Access to the IBV git source repository

Contains manifests for the different orchestrators (Docker-compose, Kubernetes, OpenShift), scripts, and configuration templates.

Turn server

This is an optional component that can be installed. This component that will allow client-to-client communication within Collaboard even across traversal of NAT networks .The following link explain more technical details Turn server is configured to run on default ports, which are 3478 for http and 5349 for https. SSL certificate needs to be provided for https connection to work properly. Certificate needs to be provided as two files, **cb-turn.crt** and **cb-turn.key**, that need to be mounted in folder **/var/turnserver/certs**. User and password needs to be specified in environmental variables TURN_USER and TURN_PSWD for the container. These can be anything, there are no restrictions. This user will be used to authenticate with the TURN server. Realm and server need to be set to the domain where the server is running. Example.: collaboard.app TURN_MIN_PORT and TURN_MAX_PORT are optional variables that default to 49152-65535 UDP port range. **Important Note**: It requires a lot of ports to be opened on your firewall or load balancer and therefor it is recommended to install it on a dedicated docker instance isolated from your orchestrator. The customer will be responsible to configure these security settings. **Configured in** : *turnserver.conf ; env.Override.js ; .env*

YouTube

In Collaboard it is possible to work with Youtube videos. To enable the functionality we need a YouTube API key, here you can find the official documentation on how to create it **Configured in** : *env.Override.js*

Zoom

It is possible to integrate Zoom meetings in Collaboard. To enable the functionality we need a Zoom API key/secret, here you can find the official documentation on how to create it **Configured in** : *env.Override.js ; ConfigurationMigration.sql*

Bing

It is possible to integrate the Bing image search in Collaboard. To enable the functionality we need a Bing Search API key, here you can find the official documentation on how to create it **Configured in** : *env.Override.js*

Help page

This is an external link to any site (public or private) where the customer can store information on how to use the application, manuals, and more. The fracture can be disabled **Configured in** : *env.Override.js*

Important note about the reliability

It is important that the customer will provide us with the right

There are four important aspects to consider

- Fault tolerance
- Replica
- Backup
- Disaster recovery

The application can work with or without any of these. It is a customer's task to understand how important are the data stored in Collaboard and apply the needed policies so that the application can keep working: - If part of the cluster (or all) goes down (fault tolerance)

- If a vital component goes down (DB, Redis, Storage) (replica)
- If some data got lost (backup)
- If the datacenter where the cluster is hosted face a calamity (disaster recovery)

Check list

To make the installation process smoother the possible it is important that you start collecting the required information and provide them to the IBV DevOps engineers the sooner possible. You may find useful the following checklist

Requirement	Status
DNS Names	
SSL certificates	
Authentication Providers	
Security	
Licensing	
User to be admin	
Email provider & templates	
Storage	

Requirement	Status
Database	
Redis Cache	
Azure CLI	
Docker Desktop	
Access to containers registry	
Access to the repository	
Turn server	
YouTube	
Zoom	
Bing	
Help page	

[[*TOC*]]

Overview

To configure application, PORTS, VARIABLES and/or MOUNTS are used to provide containers with override files, configuration script or runtime values.

Overrides

Containers are being distributed as images with a default/incomplete configuration. All the environmental values are provided using the Environment variables, override files and ConfigurationMigration.sql script.

Overrides are applied on top of the default configuration that is present in the container images. Overrides examples and templates are provided via Distribution repository. They are continuously updated and can be used as a starting point for new installation or a template for changes to existing installation. Any breaking changes are documented and sent through proper channels.

All the overrides are applied on startup of the containers. Any changes made to those files will require restart of container instances for changes to be picked up. ConfigurationMigration.sql script is applied every time the cb_database_init container is executed. After database structure is validated and updated as necessary, SQL script is always executed as last step.

List of all VARIABLES

- **SQL_SCRIPT_FILE** -- filename of SQL configuration script.
- **SA_PASSWORD** -- password for sa user of the database.
- **SA_USERNAME** -- optional, DO NOT CHANGE IF USING CB_DB CONTAINER, defaults to "sa".
- **SQL_DATABASE** -- optional, sets the name of database to create/update, defaults to IBV.Database.
- **SQL_SERVER** -- optional, specifies server where database runs, defaults to "db".
- **ACCEPT_EULA** -- required variable by DB container that needs to be set to Y - means you accept the EULA of inner SQL server.
- **MSSQL_PID** -- in case you will host the SQL database inside of a container, you will need to specify the SQL edition (Express, Standard, Enterprise, ...). In that case make sure you have SQL license.
- **CB_OVERRIDE_FILE** -- filename of appsettings override file.
- **CB_OVERRIDE_ENV_JS** -- filename of frontend env.js override file.
- **CB_OVERRIDE_BACKOFFICE** -- filename of backoffice env.js override file.
- **CB_OVERRIDE_ENVOY** -- filename of proxy envoy.yaml override file.
- **TURN_REALM** -- realm of the TURN server user, i.e. *collaboard.app*, overrides value in turnserver.conf if defined.
- **TURN_SERVER** -- domain name of TURN server, i.e. *collaboard.app*, overrides value in turnserver.conf if defined.
- **TURN_MIN_PORT** -- starting port of UDP port range for TURN server, overrides value in turnserver.conf if defined.
- **TURN_MAX_PORT** -- ending port of UDP port range for TURN server, overrides value in turnserver.conf if defined.
- **TURN_USER** -- user name for TURN server user, overrides value in turnserver.conf if defined.
- **TURN_PSWD** -- password for TURN server user, overrides value in turnserver.conf if defined.

List of all MOUNTS

- **/var/opt/mssql/data**
  - Files for SQL database data are saved here.
- **/var/opt/mssql/logs**
  - Files for SQL database logs are saved here.
- **/var/opt/mssql/migration_script**
  - Folder containing the **SQL_SCRIPT_FILE**.
- **/var/mft/_cbfiles_**
  - Files uploaded by users and such are saved here.
- **/var/mft/_temp_**
  - Temporary folder used by services to pass data in-between.
- **/var/overrides**
  - Folder containing files to override default configuration files mentioned in List of all VARIABLES and the **license.lic** file if provided.
- **/etc/envoy/certs**
  - Folder containing certificate files for proxy. They have to be named **cb-proxy.crt** and **cb-proxy.key**.
- **/var/turnserver/certs**
  - Folder containing certificate files for TURN server. They have to be named **cb-turn.crt** and **cb-turn.key**.

List of PORTS

- 8080 -- Proxy is listening on this port for non-secure communication.
- 8443 -- Proxy is listening on this port for secured communication - Web.
- 9443 -- Proxy is listening on this port for secured communication - Backoffice.
- 3478 & 3478/udp -- Port used by TURN server for non-secured WebRTC communication.
- 5349 & 5349/udp -- Port used by TURN server for secured WebRTC communication.
- 49152-65535/udp -- Range of UDP ports used by TURN server, configurable in List of all VARIABLES or in turnserver.conf. For non-production use, a smaller range can be defined.

Containers

Each container makes use of some of the defined mount points and/or variables. Redis and cb_realtime containers are required if scale-out is in plan (multiple instances of containers).

- cb_database_init
  - variables
    - SQL_SCRIPT_FILE
    - SA_PASSWORD
    - (SA_USERNAME)
    - (SQL_DATABASE)
    - (SQL_SERVER)
  - mounts
    - /var/opt/mssql/migration_script

- cb_db
  - variables
    - ACCEPT_EULA
    - SA_PASSWORD
    - (SQL_DATABASE)
  - mounts
    - /var/opt/mssql/data
    - /var/opt/mssql/logs

- cb_proxy
  - variables
    - CB_OVERRIDE_ENVOY
  - mounts
    - /var/overrides
    - /etc/envoy/certs
  - ports
    - 8080
    - 8443

- cb_frontend
  - variables
    - CB_OVERRIDE_ENV_JS
  - mounts
    - /var/overrides

- cb_backoffice
  - variables
    - CB_OVERRIDE_BACKOFFICE
  - mounts
    - /var/overrides

- cb_api
  - variables
    - CB_OVERRIDE_FILE
  - mounts
    - /var/mft/_cbfiles_

        - /var/mft/_temp_
        - /var/overrides

- cb_realtime
    - variables
        - CB_OVERRIDE_FILE
    - mounts
        - /var/overrides

- cb_auth
    - variables
        - CB_OVERRIDE_FILE
    - mounts
        - /var/overrides

- cb_licensing
    - variables
        - CB_OVERRIDE_FILE
    - mounts
        - /var/overrides

- cb_mftapi
    - variables
        - CB_OVERRIDE_FILE
    - mounts
        - /var/mft/_cbfiles_
        - /var/mft/_temp_
        - /var/overrides

- cb_copyworker
    - variables
        - CB_OVERRIDE_FILE
    - mounts
        - /var/overrides

- cb_worker1
    - variables
        - CB_OVERRIDE_FILE
    - mounts
        - /var/overrides

- cb_worker2
    - variables
        - CB_OVERRIDE_FILE
    - mounts
        - /var/overrides

- cb_canvasshotter
    - variables
        - CB_OVERRIDE_FILE
    - mounts
        - /var/mft/_cbfiles_
        - /var/mft/_temp_
        - /var/overrides

- mft_chunkservice
    - variables
        - CB_OVERRIDE_FILE
    - mounts
        - /var/mft/_cbfiles_
        - /var/mft/_temp_
        - /var/overrides

- mft_cleanerservice
    - variables
        - CB_OVERRIDE_FILE
    - mounts
        - /var/mft/_cbfiles_
        - /var/mft/_temp_
        - /var/overrides

- mft_mergeservice
    - variables
        - CB_OVERRIDE_FILE
    - mounts
        - /var/mft/_cbfiles_
        - /var/mft/_temp_
        - /var/overrides

- mft_storageoperationsservice
    - variables
        - CB_OVERRIDE_FILE
    - mounts
        - /var/mft/_cbfiles_
        - /var/mft/_temp_
        - /var/overrides

- cb_fileconverter
    - variables
        - CB_OVERRIDE_FILE
    - mounts
        - /var/mft/_cbfiles_
        - /var/mft/_temp_
        - /var/overrides

- cb_imageresizer
    - variables
        - CB_OVERRIDE_FILE
    - mounts
        - /var/mft/_cbfiles_
        - /var/mft/_temp_
        - /var/overrides

- cb_coturn
    - variables
        - TURN_REALM
        - TURN_SERVER
        - TURN_MIN_PORT
        - TURN_MAX_PORT
        - TURN_USER
        - TURN_PSWD
    - mounts
        - /var/overrides

        - /var/turnserver/certs
    - ports
        - 3478
        - 3478/udp
        - 5349
        - 5349/udp
        - 49152-65535/udp -- *must be aligned with TURN_MIN_PORT - TURN_MAX_PORT*

- redis
    - variables
        - *none*
    - mounts
        - *none*

[[*TOC*]]

Overview

Table ***dbo.t_configuration*** contains most of the environmental settings. On change made to this table, most of the services/APIs require rows in ***dbo.t_caches*** table to have their ***publishid*** value updated to new random GUID for hot reload. Some services/APIs require restart upon change of specific keys.

Following are the groups of values based on the Full example script at the end of this page.

1. Main URLs

These values hold the URLs used by services for internal communication inside the closed network. Unless containers domain names or routing are changed, these values will stay the same.

There is one exception in form of ***Client_AuthBaseUri*** which contains EXTERNAL URL leading to the **Authorization controller** of the **cb_auth** container.

For ease of use, shared parts of root URLs are declared as variables at start of the Full example script.

2. No REDIS / With REDIS

These two groups are two side of the same coin. Depending on whether Realtime with REDIS are used or not, one of these groups is left out.

Server_RedisCacheConfig

Connection string consists of several parts separated by comma:

- server
  - positional required parameter, [server] format
  - in case of REDIS instance running on different than default port, it needs to be specified after colon in "*server:port*" format
- ssl
  - optional parameter, ssl=[True/False] format
  - indicates whether encrypted communication should be used
- password
  - optional parameter, password=[value] format
- abortConnect
  - optional parameter, abortConnect=[True/False] format
  - specifies whether connection should be aborted on failure (prevents reconnection)

Examples:

- `redis,ssl=False,abortConnect=False`
- `redis-name.redis.cache.windows.net:6380,password=*****,ssl=True,abortConnect=False`

Server_RedisCacheServerName

Name of connection. By default REDIS server URI without port can be specified.

Examples:

- `redis`
- `redis-name.redis.cache.windows.net`

3. Security

Server_WebApiOTPSecret and Server_WebApiTokenSecret

These two keys hold random strings that are used for securing the OTP token and access/refresh token generation/encryption. The Server_WebApiOTPSecret must have a length of be 32 characters, and the Server_WebApiTokenSecret must have a length of 64 characters. In the Full example script random string values have been used.

Server_CryptextKeys

The value of this setting is a JSON object, serialized as a string, that holds the key and initialization vector (IV) properties for the symmetric encryption algorithm. The key must be 32 bytes and the IV must be 16 bytes.

```
{
  "Key": "[[SYMMETRIC ENCRYPTION KEY, AS A BASE64 STRING]]",
  "IV": "[[INITIALIZATION VECTOR, AS A BASE64 STRING]]"
}
```

Server_WebApiExternalLoginProviders

Value of this configuration key is a JSON array of objects representing sets of configurations. Most of the time there will be only one set of providers defined.

The **AppCode** value of the set is it's ID and based on value specified in **appName** of env.Override.js

Following is the template for all 5 currently supported providers. Template variables (text enclosed in [[ ]]) need to be replaced with proper values (including the [[ ]]).

- [[CLIENT_ID]] -- Client ID of the app registration.
- [[CLIENT_SECRET]] -- Client Secret generated for the app registration.
- [[WEB_APP_URL]] -- Base URL under which frontend application is accessible. (i.e. web.collaboard.app)
- [[ADFS_URL]] -- ADFS server URL.
- [[SAML_AUTH_URL]] -- SAML Identity Provider authenticate URL.
- [[X509_CERTIFICATE]] -- SAML Identity Provider public X509 certificate
- [[WEB_API_URL]] -- Base URL under which web api is accessible. (i.e. api.collaboard.app)

```
[
    {
        "AppCode": "webapp",
        "LoginProviders": [
            {
                "Provider": "ADFS",
                "ClientId": "[[CLIENT_ID]]",
                "ClientSecret": "[[CLIENT_SECRET]]",
                "RedirectUri": "https://[[WEB_APP_URL]]/authenticate",
                "AuthorizeUri": "https://[[ADFS_URL]]/adfs/oauth2/authorize",
                "TokenUri": "https://[[ADFS_URL]]/adfs/oauth2/token",
                "Scope": "openid email profile",
                "TFAEnabled": false
            },
            {
                "Provider": "SAML",
                "ClientId": "[[CLIENT_ID]]",
                "ClientSecret": "[[X509_CERTIFICATE]]",
                "RedirectUri": "https://[[WEB_API_URL]]/server/auth/api/Authorization/HandleExternalLoginCallback?app=webapp&provider=SAML&redirectUri=https%3A%2F%2F[[WEB_APP_URL
                "AuthorizeUri": "[[SAML_AUTH_URL]]",
                "TokenUri": null,
                "Scope": null,
                "TFAEnabled": false
            },
            {
                "Provider": "Google",
                "ClientId": "[[CLIENT_ID]]",
                "ClientSecret": "[[CLIENT_SECRET]]",
                "AuthorizeUri": "https://accounts.google.com/o/oauth2/v2/auth",
                "TokenUri": "https://www.googleapis.com/oauth2/v4/token",
                "RedirectUri": "https://[[WEB_APP_URL]]/authenticate",
                "Scope": "openid email profile",
                "TFAEnabled": false
            },
            {
                "Provider": "Microsoft",
                "ClientId": "[[CLIENT_ID]]",
                "ClientSecret": "[[CLIENT_SECRET]]",
                "AuthorizeUri": "https://login.microsoftonline.com/common/oauth2/v2.0/authorize",
                "TokenUri": "https://login.microsoftonline.com/common/oauth2/v2.0/token",
                "RedirectUri": "https://[[WEB_APP_URL]]/authenticate",
                "Scope": "openid email profile",
                "TFAEnabled": false
            },
            {
                "Provider": "Apple",
```

```
                "ClientId": "[[CLIENT_ID]]",
                "ClientSecret": "[[CLIENT_SECRET]]",
                "AuthorizeUri": "https://appleid.apple.com/auth/authorize",
                "TokenUri": "https://appleid.apple.com/auth/token",
                "RedirectUri": "https://[[WEB_API_URL]]/server/auth/api/Authorization/HandleExternalLoginCallback?app=webapp&provider=Apple&redirectUri=https%3A%2F%2F[[WEB_APP_URI
                "Scope": "name email",
                "TFAEnabled": false
            }
        ]
    }
]
```

4. Email

Email configuration. Values are set from Planning the installation step.

5. MFT

This section contains MFT related values.

Client_MFTServiceBaseUri and Server_MftUri

Those two keys are to be set to the same value. They contain closed network address of MFT API container.

Server_RemoteStorageType

Indicates what kind of storage is used. Currently only two values are supported, 0 for MFT and 1 for Azure blob storage.

Server_BaseOnPremisePath

Network path leading to where Collaboard data are being persisted. It needs to be aligned with respective mount point (see mount points here).

TempPath

Network path leading to where temp files are being shared passed along among services. It needs to be aligned with respective mount point (see mount points here).

6. Licensing

Here you will provide path to the mounted licensing file, for example '/var/overrides/license.lic'.

7. ZOOM

If you are going to use ZOOM with your installation, you need to provide ApiKey and ApiSecret for it.

Full template

```
USE [IBV.Database]
GO

DECLARE @externalUrl NVARCHAR(50) = '< EXTERNAL URL >'
DECLARE @internalUrl NVARCHAR(50) =  'http://api:8080'
DECLARE @internalMftUrl NVARCHAR(50) = 'http://mftapi:8080'
DECLARE @internalAuthUrl NVARCHAR(50) = 'http://auth:8080'
DECLARE @internalLicensingUrl NVARCHAR(50) = 'http://licensing:8080'

---- With REDIS + Realtime
DECLARE @realtimeUrl NVARCHAR(50) =  'http://realtime:8080'

DECLARE @DefaultAppDomain NVARCHAR(50) = 'CollaboardServices'
DECLARE @GenericAppDomain NVARCHAR(50) = '*'


DECLARE @Configuration TABLE ([key] NVARCHAR(35), [value] NVARCHAR(MAX), [description] NVARCHAR(250), [AppDomain] NVARCHAR(50))

INSERT INTO @Configuration
VALUES ('Client_HubServiceBaseUri', @internalUrl + '/api/collaborationhub', 'Base uri for Hub Service',  @DefaultAppDomain)
,('Client_AuthBaseUri', @externalUrl + '/server/auth/api/Authorization', 'The URI for the authorization systems',  @GenericAppDomain)
,('Server_AuthBaseUri', @internalAuthUrl + '/api/authorization', 'The URI for the authorization systems',  @GenericAppDomain)
,('Client_CfgServiceBaseUri', @internalUrl + '/api/clientconfiguration', 'URI for the client configuration',  @DefaultAppDomain)
,('Server_CanvasShotterURL', @externalUrl + '/collaboard/{0}/{1}/{2}','', @DefaultAppDomain)

---- No REDIS
--,('Client_HubUrl', @internalUrl + '/signalr', 'SignalR path',  @GenericAppDomain)
--,('Server_RedisCacheEnabled', 'false', 'Indicates if Redis cache is enabled or not', @DefaultAppDomain)

---- With REDIS + Realtime
,('Client_HubUrl', @realtimeUrl + '/signalr', 'SignalR path',  @GenericAppDomain)
,('Server_RedisCacheEnabled', 'true', 'Indicates if Redis cache is enabled or not', @DefaultAppDomain)
,('Server_RedisCacheConfig', '< PROVIDE VALUE >', 'Redis Cache Configuration', @DefaultAppDomain)
,('Server_RedisCacheServerName', '< PROVIDE VALUE >', 'Server Name fro Redis Cache', @DefaultAppDomain)

---- Security
,('Server_WebApiOTPSecret', '< PROVIDE VALUE >', 'Server Web API OTP generation security key',  @GenericAppDomain)
,('Server_WebApiOTPDurationMins', '< PROVIDE VALUE >', 'Server Web API generated OTP code duration in minutes', @GenericAppDomain)
,('Server_WebApiTokenSecret', '< PROVIDE VALUE >', 'Server Web API authentication token encryption secret key',  @GenericAppDomain)
,('Server_WebApiRefrTokenDurationMins', '< PROVIDE VALUE >', 'Server Web API refresh token duration in minutes', @GenericAppDomain)
,('Server_WebApiAuthTokenDurationMins', '< PROVIDE VALUE >', 'Server Web API authentication token duration in minutes', @GenericAppDomain)
,('Server_WebApiExternalLoginProviders', '< PROVIDE VALUE >', 'External authentication providers configuration',  @GenericAppDomain)
,('Server_CryptextKeys', '< PROVIDE VALUE >', 'Key and IV for symmetric encrypting and decrypting', @GenericAppDomain)

--- Email
,('Server_MailProviderConnectionString', '< PROVIDE VALUE >', 'Mail provider connection string', @DefaultAppDomain)
,('Server_MailTemplates', '[{
  "Theme": "default",
  "Templates": [
    { "Code": "ResetUserPassword", "Language": "en-US", "Sender": "noreply@collaboard.app", "Recipient": "{Recipient}", "Subject": "We got a request to reset your Collaboard passw
    { "Code": "VerifyUserAccount", "Language": "en-US", "Sender": "noreply@collaboard.app", "Recipient": "{Recipient}", "Subject": "Please verify your account", "Body": "<!DOCTYPE
    { "Code": "SendUserOTP", "Language": "en-US", "Sender": "noreply@collaboard.app", "Recipient": "{Recipient}", "Subject": "Here is your one-time password", "Body": "<!DOCTYPE h
    { "Code": "InviteProjectParticipant", "Language": "en-US", "Sender": "noreply@collaboard.app", "Recipient": "{Recipient}", "Subject": "You have been invited to a Collaboard pr
    { "Code": "ImportLicensedUser", "Language": "en-US", "Sender": "noreply@collaboard.app", "Recipient": "{Recipient}", "Subject": "Collaboard license activated", "Body": "<!DOCT
  ]
 }]', 'Mail templates for email sending', @DefaultAppDomain)

---- MFT Storage
,('Server_RemoteStorageType', '0', '0 = OnPrem, 1 = Azure',  @GenericAppDomain)
,('Client_MFTServiceBaseUri', @internalMftUrl, 'Uri for the MFT Service',  @GenericAppDomain)
,('Server_MftUri', @internalMftUrl, 'Uri for the MFT Service',  @GenericAppDomain)
,('Server_BaseOnPremisePath', '/var/mft/_cbfiles_', 'Base path to store project files',  @DefaultAppDomain)
,('TempPath', '/var/mft/_temp_', 'Temporary path for storing files',  @GenericAppDomain)

---- Licensing
,('Client_LicensingServiceBaseUri', @internalLicensingUrl + '/api/licensing', 'Uri for the licensing services',  @GenericAppDomain)
,('Server_LicensingLicenseProvider', 'Provider=FileLicenseProvider;LicenseFile=< PATH TO MOUNTED LICENSE.LIC FILE >;', 'Licensing - License provider connection string', @DefaultAp
,('Server_LicensingPlans', '[{"Name":"Free","ProductId":null,"NumberOfProjects":3,"NumberOfParticipants":5,"GuestParticipantsAllowed":false},{"Name":"Personal","ProductId":1,"Numb

---- ZOOM
,('Server_Zoom_ApiKey', '< PROVIDE VALUE >', 'API Key for connecting to the Zoom server', @DefaultAppDomain)
,('Server_Zoom_ApiSecret', '< PROVIDE VALUE >', 'API Secret for connecting to the Zoom server', @DefaultAppDomain)



DECLARE @Key NVARCHAR(35)
DECLARE @Value NVARCHAR(MAX)
DECLARE @Description NVARCHAR(250)
DECLARE @AppDomain NVARCHAR(50)

WHILE EXISTS(SELECT TOP (1) 1 FROM @Configuration)
BEGIN
 SELECT TOP (1) @Key = [key], @Value = [value], @Description = [description], @AppDomain = [AppDomain] FROM @Configuration
```

```
 IF EXISTS(SELECT TOP (1) 1 FROM [dbo].[t_configuration] WHERE [key] = @Key)
 BEGIN
  UPDATE
   [dbo].[t_configuration]
  SET
   [value] = @Value,
   [description] = @Description,
   [AppDomain] = @AppDomain
  WHERE
   [key] = @Key
 END
 ELSE
 BEGIN
  INSERT INTO
   [dbo].[t_configuration] ([key], [value], [description], [AppDomain])
  VALUES
   (@Key, @Value, @Description, @AppDomain)
 END

 DELETE FROM
  @Configuration
 WHERE
  [key] = @Key
END

-- Delete all caches, insert new values so we are sure it all works just fine now...
DELETE from dbo.t_caches

INSERT INTO dbo.t_caches
(
    cacheid,
    publishid,
    pollingtimeinseconds,
    cachename,
    cachedescription
)
VALUES
( 0, NEWID(), 300, N'cache', N'cache for cache' ),
( 1, NEWID(), 300, N'configuration', N'cache for configuration' )
```

[[*TOC*]]

Overview

ENVOY override file is used in case of differences in routing. File as a whole describes how requests are routed to respective containers. On startup, proxy container will read the configuration. Then it attempts to resolve all defined routes. In case it fails to resolve any of the used paths, it will still work but cb_proxy startup will hand until the internal check times out.

Presence of Realtime

Whether Realtime is being used in the installation, the configuration needs to reflect that. It boils down to presence of following code snippet in clusters definition:

```
static_resources:
  listeners:
  - name: main_listener
    ...
    filter_chains:
    - filters:
      - name: envoy.filters.network.http_connection_manager
        typed_config:
          ...
          route_config:
            name: local_route
            virtual_hosts:
            - name: main_routing
              ...
              routes:
                - match:
                    prefix: "/server/realtime/signalr"
                  route:
                    prefix_rewrite: "/signalr"
                    cluster: realtime_cluster
                    timeout: 300s
                    upgrade_configs:
                      upgrade_type: "websocket"
                      enabled: true
  ...
```

```
static_resources:
  ...
  clusters:
  - name: realtime_cluster
    connect_timeout: 30s
    type: LOGICAL_DNS
    dns_lookup_family: V4_ONLY
    lb_policy: round_robin
    load_assignment:
      cluster_name: no1
      endpoints:
      - lb_endpoints:
        - endpoint:
            address:
              socket_address:
                address: realtime
                port_value: 8080
  ...
```

And main

Container names change

Platform hosting the containers might require change of name under which the container is accessible in the closed network. In that case, respective values for **address** must be changed to new names. For example AWS ECS is enforcing for names to be compliant with URL format containing first and second level domain names.

Full example

Following are two examples.

1. Localhost with Realtime

Backoffice routing is port based, separate listener is used for it.

External addresses: ||| -|- Frontend | https://localhost:8443 Backoffice | https://localhost:9443

Requests lading on port 8080 of cb_proxy is redirected to the outside address with port 8443.

Example:

```
static_resources:
  listeners:
  - name: main_listener
    address:
      socket_address:
        address: 0.0.0.0
        port_value: 8443
    listener_filters:
    - name: "envoy.filters.listener.tls_inspector"
      typed_config: {}
    filter_chains:
    - filters:
      - name: envoy.filters.network.http_connection_manager
        typed_config:
          "@type": type.googleapis.com/envoy.extensions.filters.network.http_connection_manager.v3.HttpConnectionManager
          stat_prefix: ingress_http
          http_filters:
          - name: envoy.filters.http.router
          route_config:
            name: local_route
            response_headers_to_add:
              - header:
                  key: "X-XSS-Protection"
                  value: "1"
                append: true
              - header:
                  key: "X-Content-Type-Options"
                  value: "nosniff"
                append: true
              - header:
                  key: "Strict-Transport-Security"
                  value: "max-age=3600; includeSubDomains"
                append: true
              - header:
                  key: "Content-Security-Policy"
                  value: "frame-ancestors 'self' teams.microsoft.com *.teams.microsoft.com *.skype.com"
                append: true
              - header:
                  key: "X-Frame-Options"
                  value: "DENY"
                append: true
            virtual_hosts:
            - name: main_routing
              domains: ["*"]
              routes:
                - match:
                    prefix: "/server/realtime/signalr"
                  route:
                    prefix_rewrite: "/signalr"
                    cluster: realtime_cluster
                    timeout: 300s
```

```
                  upgrade_configs:
                    upgrade_type: "websocket"
                    enabled: true
              - match:
                  prefix: "/server/auth/api/"
                route:
                  prefix_rewrite: "/api/"
                  cluster: auth_cluster
              - match:
                  prefix: "/server/licensing/api/"
                route:
                  prefix_rewrite: "/api/"
                  cluster: licensing_cluster
              - match:
                  prefix: "/server/api/"
                route:
                  prefix_rewrite: "/api/"
                  cluster: backend_cluster
              - match:
                  prefix: "/mft/api/"
                route:
                  prefix_rewrite: "/api/"
                  cluster: mftapi_cluster
              - match:
                  prefix: "/"
                route:
                  cluster: frontend_cluster
    transport_socket:
      name: envoy.transport_sockets.tls
      typed_config:
        "@type": type.googleapis.com/envoy.extensions.transport_sockets.tls.v3.DownstreamTlsContext
        common_tls_context:
          tls_certificates:
            certificate_chain:
              filename: /etc/envoy/certs/cb-proxy.crt
            private_key:
              filename: /etc/envoy/certs/cb-proxy.key
          alpn_protocols: [ "h2,http/1.1" ]

- name: admin_listener
  address:
    socket_address:
      address: 0.0.0.0
      port_value: 9443
  listener_filters:
  - name: "envoy.filters.listener.tls_inspector"
    typed_config: {}
  filter_chains:
  - filters:
    - name: envoy.filters.network.http_connection_manager
      typed_config:
        "@type": type.googleapis.com/envoy.extensions.filters.network.http_connection_manager.v3.HttpConnectionManager
        stat_prefix: ingress_http
        http_filters:
        - name: envoy.filters.http.router
        route_config:
          name: local_route
          virtual_hosts:
          - name: main_routing
            domains: ["*"]
            routes:
              - match:
                  prefix: "/"
                route:
                  cluster: backoffice_cluster
    transport_socket:
      name: envoy.transport_sockets.tls
      typed_config:
        "@type": type.googleapis.com/envoy.extensions.transport_sockets.tls.v3.DownstreamTlsContext
        common_tls_context:
          tls_certificates:
            certificate_chain:
              filename: /etc/envoy/certs/cb-proxy.crt
            private_key:
              filename: /etc/envoy/certs/cb-proxy.key
          alpn_protocols: [ "h2,http/1.1" ]

- name: 8080_redirect
  address:
    socket_address:
      address: 0.0.0.0
      port_value: 8080
  filter_chains:
  - filters:
    - name: envoy.filters.network.http_connection_manager
      typed_config:
        "@type": type.googleapis.com/envoy.extensions.filters.network.http_connection_manager.v3.HttpConnectionManager
        stat_prefix: ingress_http
        http_filters:
        - name: envoy.filters.http.router
        route_config:
          name: local_route
          virtual_hosts:
          - name: http_redirect_routing
            domains: ["*"]
            routes:
              - match:
                  prefix: "/"
                redirect:
                  https_redirect: true
                  port_redirect: 8443

  clusters:
  - name: frontend_cluster
    connect_timeout: 30s
    type: LOGICAL_DNS
    dns_lookup_family: V4_ONLY
    lb_policy: round_robin
    load_assignment:
      cluster_name: no1
      endpoints:
      - lb_endpoints:
        - endpoint:
            address:
              socket_address:
                address: frontend
                port_value: 8080
  - name: backoffice_cluster
    connect_timeout: 30s
    type: LOGICAL_DNS
    dns_lookup_family: V4_ONLY
    lb_policy: round_robin
    load_assignment:
      cluster_name: no1
      endpoints:
      - lb_endpoints:
        - endpoint:
            address:
              socket_address:
                address: backoffice
```

```
              port_value: 8080
  - name: backend_cluster
    connect_timeout: 30s
    type: LOGICAL_DNS
    dns_lookup_family: V4_ONLY
    lb_policy: round_robin
    load_assignment:
      cluster_name: no1
      endpoints:
      - lb_endpoints:
        - endpoint:
            address:
              socket_address:
                address: api
                port_value: 8080
  - name: realtime_cluster
    connect_timeout: 30s
    type: LOGICAL_DNS
    dns_lookup_family: V4_ONLY
    lb_policy: round_robin
    load_assignment:
      cluster_name: no1
      endpoints:
      - lb_endpoints:
        - endpoint:
            address:
              socket_address:
                address: realtime
                port_value: 8080
  - name: auth_cluster
    connect_timeout: 30s
    type: LOGICAL_DNS
    dns_lookup_family: V4_ONLY
    lb_policy: round_robin
    load_assignment:
      cluster_name: no1
      endpoints:
      - lb_endpoints:
        - endpoint:
            address:
              socket_address:
                address: auth
                port_value: 8080
  - name: licensing_cluster
    connect_timeout: 30s
    type: LOGICAL_DNS
    dns_lookup_family: V4_ONLY
    lb_policy: round_robin
    load_assignment:
      cluster_name: no1
      endpoints:
      - lb_endpoints:
        - endpoint:
            address:
              socket_address:
                address: licensing
                port_value: 8080
  - name: mftapi_cluster
    connect_timeout: 30s
    type: LOGICAL_DNS
    dns_lookup_family: V4_ONLY
    lb_policy: round_robin
    load_assignment:
      cluster_name: no1
      endpoints:
      - lb_endpoints:
        - endpoint:
            address:
              socket_address:
                address: mftapi
                port_value: 8080
```

2. Changed name example

Example for AWS ECS which contains part where change for http://api to http://api.ibvlocal.cl2 is reflected. Backoffice routing is domain based.

External addresses: ||| -|- Frontend | https://ecs-cl2.example.com Backoffice | https://ecs-cl2-admin.example.com

Requests lading on port 8080 of cb_proxy is redirected to the outside address with port 443.

Example:

```
static_resources:
  listeners:
  - name: main_listener
    address:
      socket_address:
        address: 0.0.0.0
        port_value: 443
    listener_filters:
    - name: "envoy.filters.listener.tls_inspector"
      typed_config: {}
    filter_chains:
    - filters:
      - name: envoy.filters.network.http_connection_manager
        typed_config:
          "@type": type.googleapis.com/envoy.extensions.filters.network.http_connection_manager.v3.HttpConnectionManager
          stat_prefix: ingress_http
          http_filters:
          - name: envoy.filters.http.router
          route_config:
            name: local_route
            response_headers_to_add:
              - header:
                  key: "X-XSS-Protection"
                  value: "1"
                append: true
              - header:
                  key: "X-Content-Type-Options"
                  value: "nosniff"
                append: true
              - header:
                  key: "Strict-Transport-Security"
                  value: "max-age=3600; includeSubDomains"
                append: true
              - header:
                  key: "Content-Security-Policy"
                  value: "frame-ancestors 'self' teams.microsoft.com *.teams.microsoft.com *.skype.com"
                append: true
              - header:
                  key: "X-Frame-Options"
                  value: "DENY"
                append: true
            virtual_hosts:
            - name: admin_routing
              domains: ["ecs-cl2-admin.*"]
              routes:
                - match:
                    prefix: "/"
                  route:
                    cluster: backoffice_cluster
```

```
                  auto_host_rewrite: true
          - name: main_routing
            domains: ["*"]
            routes:
              - match:
                  prefix: "/server/realtime/signalr"
                route:
                  prefix_rewrite: "/signalr"
                  cluster: realtime_cluster
                  auto_host_rewrite: true
                  timeout: 300s
                  upgrade_configs:
                    upgrade_type: "websocket"
                    enabled: true
              - match:
                  prefix: "/server/auth/api/"
                route:
                  prefix_rewrite: "/api/"
                  cluster: auth_cluster
                  auto_host_rewrite: true
              - match:
                  prefix: "/server/licensing/api/"
                route:
                  prefix_rewrite: "/api/"
                  cluster: licensing_cluster
                  auto_host_rewrite: true
              - match:
                  prefix: "/mft/api/"
                route:
                  prefix_rewrite: "/api/"
                  cluster: mftapi_cluster
                  auto_host_rewrite: true
              - match:
                  prefix: "/server/api/"
                route:
                  prefix_rewrite: "/api/"
                  cluster: backend_cluster
                  auto_host_rewrite: true
              - match:
                  prefix: "/"
                route:
                  cluster: frontend_cluster
                  auto_host_rewrite: true
    transport_socket:
      name: envoy.transport_sockets.tls
      typed_config:
        "@type": type.googleapis.com/envoy.extensions.transport_sockets.tls.v3.DownstreamTlsContext
        common_tls_context:
          tls_certificates:
            certificate_chain:
              filename: /etc/envoy/certs/cb-proxy.crt
            private_key:
              filename: /etc/envoy/certs/cb-proxy.key
          alpn_protocols: [ "h2,http/1.1" ]

clusters:
- name: backend_cluster
  connect_timeout: 30s
  type: LOGICAL_DNS
  dns_lookup_family: V4_ONLY
  lb_policy: round_robin
  load_assignment:
    cluster_name: no1
    endpoints:
    - lb_endpoints:
      - endpoint:
          hostname: api.ibvlocal.cl2
          address:
            socket_address:
              address: api.ibvlocal.cl2
              port_value: 8080
- name: frontend_cluster
  connect_timeout: 30s
  type: LOGICAL_DNS
  dns_lookup_family: V4_ONLY
  lb_policy: round_robin
  load_assignment:
    cluster_name: no1
    endpoints:
    - lb_endpoints:
      - endpoint:
          hostname: frontend.ibvlocal.cl2
          address:
            socket_address:
              address: frontend.ibvlocal.cl2
              port_value: 8080
- name: backoffice_cluster
  connect_timeout: 30s
  type: LOGICAL_DNS
  dns_lookup_family: V4_ONLY
  lb_policy: round_robin
  load_assignment:
    cluster_name: no1
    endpoints:
    - lb_endpoints:
      - endpoint:
          hostname: backoffice.ibvlocal.cl2
          address:
            socket_address:
              address: backoffice.ibvlocal.cl2
              port_value: 8080
# With REDIS + Realtime
- name: realtime_cluster
  connect_timeout: 30s
  type: LOGICAL_DNS
  dns_lookup_family: V4_ONLY
  lb_policy: round_robin
  load_assignment:
    cluster_name: no1
    endpoints:
    - lb_endpoints:
      - endpoint:
          hostname: realtime.ibvlocal.cl2
          address:
            socket_address:
              address: realtime.ibvlocal.cl2
              port_value: 8080
- name: auth_cluster
  connect_timeout: 30s
  type: LOGICAL_DNS
  dns_lookup_family: V4_ONLY
  lb_policy: round_robin
  load_assignment:
    cluster_name: no1
    endpoints:
    - lb_endpoints:
      - endpoint:
          hostname: auth.ibvlocal.cl2
          address:
            socket_address:
```

```
              address: auth.ibvlocal.cl2
              port_value: 8080
  - name: licensing_cluster
    connect_timeout: 30s
    type: LOGICAL_DNS
    dns_lookup_family: V4_ONLY
    lb_policy: round_robin
    load_assignment:
      cluster_name: no1
      endpoints:
      - lb_endpoints:
        - endpoint:
            hostname: licensing.ibvlocal.cl2
            address:
              socket_address:
                address: licensing.ibvlocal.cl2
                port_value: 8080
  - name: mftapi_cluster
    connect_timeout: 30s
    type: LOGICAL_DNS
    dns_lookup_family: V4_ONLY
    lb_policy: round_robin
    load_assignment:
      cluster_name: no1
      endpoints:
      - lb_endpoints:
        - endpoint:
            hostname: mftapi.ibvlocal.cl2
            address:
              socket_address:
                address: mftapi.ibvlocal.cl2
                port_value: 8080
```

[[*TOC*]]

Overview

File contains configuration for the front end application. It is in a form of JS file, which initializes runtimeConfig object with appropriate values on browser root. File is replacing the original one, there is no override logic.

Structure

The file contains three parts:

- Configuration values on root level.
- **features** object containing feature flags.
- **webRTC** object containing TURN server connection configuration.

Main points

Following are the most important keys when it comes to environment configuration.

apiUrl

URL of the API. Most probably the "localhost:8443" will need to be replaced.

signalRPath

Relative URL leading to the SignalR endpoint. Here it depends whether Realtime with REDIS is used. It is prepended with **apiUrl** value on use. The full example contains commented option with or without REDIS.

appName

This value defines the name of the external authentication providers set defined in the database. On change, respective set of providers needs to be defined. More here.

authProviders

An array of supported authentication providers. For each one specified, respective definition must exist in database configuration. More here.

isOnPremise

If **MFT** is being used as file storage provider, this needs to be set to true.

onPremiseMftApiUrl

URL of the **MFT Web API**, which is used in case of **isOnPremise** being set to **true**.

webRTC

Configuration of WebRTC server (STUN/TURN) used for mouse movement sharing while presenting.

- **stunTurnUrl** -- The URL of the stun server.
- **username** -- The username of the TURN user used for connection (single user per environment).
- **credential** -- The password of the TURN user.
- **iceTransportPolicy** -- Transport policy, should stay set to **relay**.

Full example

```
window.runtimeConfig = {
  apiUrl: "https://localhost:8443/server",
  apiTimeout: 20000,
  signalRLogging: false,

  signalRPath: "/realtime/signalr",

  appName: "webapp",
  env: "production",
  isOnPremise: true,
  onPremiseMftApiUrl: "https://localhost:8443/mft",
  helpPageUrl: "https://help.collaboard.app/",
  stripePublishableKey: "",
  youtubeApiKey: "",
  customer: "default",
  authProviders: ["Google", "Microsoft"],
  features: {
    areaSelection: true,
    autoAlignment: true,
    batchUserImport: true,
    canvasMouseTracker: true,
    clickableURLs: true,
    comments: false,
    connectionToInk: false,
    copyPaste: true,
    deviceLicenses: false,
    disableConsoleLogs: true,
    documents: true,
    embed: true,
    embedFonts: false,
    generateProjectThumbnail: true,
    imageCrop: false,
    inkSplitting: false,
    inkV2: true,
    licensing: true,
    loginPageHelpButton: true,
    minimap: true,
    panOnSpace: true,
    plansPage: true,
    presentationMode: true,
    projectListV2: true,
    selectionMode: true,
    selectObjectInGroup: true,
    shareModal: true,
    signalR: true,
    stripeAntiFraud: true,
    suppressErrorsInMSTeams: true,
    teamPage: true,
    templates: true,
    testUtils: false,
    textInShape: true,
    useCompositeTogetherWithInk: false,
    userPresence: true,
    videoCrop: false,
    videos: true,
    voting: true,
    zoomMeeting: true,
    zoomWarning: true,
    storageClient: true,
    disableProjectKey: true,
    createProjectToS: true,
    noEcommerce: true,
  },
  maxSignalRMessageSizeInKB: 64,
  bigInkLowerBoundary: 500,
  bigInkUpperBoundary: 1500,
  useCacheForConnectionAnchor: true,
  checkForNewVersionInterval: 0, // minutes
  autoAlignZoomRatio: 50, // the size zoom ratio for which auto alignment is enabled
  autoAlignSnapRadius: 3, // in this radius the objects will snap to each other // smoothness of the auto align
  monitoringSignalRQueueGradientThreshold: 1,
  timerForEditableObjectBorder: 1500,
  maxAnimatedObjectsNumber: 15,
  webRTC: {
    stunTurnUrl: "localhost:3478",
    username: "turn-user",
```

```
    credential: "aa5c1894dfs",
    iceTransportPolicy: "relay"
  }
};
```

[[*TOC*]]

Overview

File contains configuration for the backoffice application. It is in a form of JS file, which initializes runtimeConfig object with appropriate values on browser root. File is replacing the original one, there is no override logic.

Main points

Following are the most important keys when it comes to environment configuration.

apiUrl

URL of the API. For example https://web.example.com/server or https://api.example.com depending on your configuration.

appName

This value defines the name of the external authentication providers set defined in the database. On change, respective set of providers needs to be defined. More here.

authProviders

An array of supported authentication providers. For each one specified, respective definition must exist in database configuration. More here.

Full example

```
window.runtimeConfig = {
    apiUrl: "https://localhost:8443/server",
    appName: "webapp-admin",
    authProviders: ["Google", "Microsoft"],
    displayErrorDetails: false,
    pages:{
      messageTemplates: true,
      onlineUsers: true,
      products: true,
      projects: true,
      projectTiles: true,
      subscriptionUsers: true,
      templateCategories: true,
      templateProjects: true,
      users: true,
    }
  }
```

[[*TOC*]]

Overview

The variable CB_OVERRIDE_FILE points to filename of appsettings override file in **JSON** format. Usually the filename is called **appsettings.Override.json**.

Main purpose of this file is to change database connection strings, allowed URLs and Logging. All other configuration is done through the dbo.t_configuration table described in script file documentation.

Unless there is need for different values for each containers due to logging settings, the file can be shared among containers that require this file.

Structure

File can be broken down to following sections.

1. ConnectionStrings

```
...

    "ConnectionStrings": {
      "CollaborationHubDB": "Server=db,1433;Initial Catalog=IBV.Database;Persist Security Info=False;User ID=sa;Password=mysecret123;Integrated Security=false;Connection Timeout=
      "IdentityDB": "Server=db,1433;Initial Catalog=IBV.Database;Persist Security Info=False;User ID=sa;Password=mysecret123;Integrated Security=false;Connection Timeout=30;",
      "ConfigurationDB": "Server=db,1433;Initial Catalog=IBV.Database;Persist Security Info=False;User ID=sa;Password=mysecret123;Integrated Security=false;Connection Timeout=30;
      "LicensingDB": "Server=db,1433;Initial Catalog=IBV.Database;Persist Security Info=False;User ID=sa;Password=mysecret123;Integrated Security=false;Connection Timeout=30;",
      "MFT_DB": "Server=db,1433;Initial Catalog=IBV.Database;Persist Security Info=False;User ID=sa;Password=mysecret123;Integrated Security=false;Connection Timeout=30;"
    }

...
```

2. Cors (Cross-origin resource sharing)

In this section we list allowed URLs that are allowed to access the application. Attempt from any other URL will result in cross-site origin error.

```
...

  "Cors": {
    "AllowedOrigins": [
      "https://localhost:8443",
      "chrome-search://local-ntp"
    ]
  }

...
```

3. Logging

There are multiple levels of logging going from lowest level (Trace, least important) to highest (Error, exceptions and similar only). The lower the level, the more information is logged.

Allowed values: **Trace**, **Debug**, **Information**, **Warning**, **Error**

```
...

  "Logging": {
    "LogLevel": {
      "Default": "Error",
      "Microsoft": "Error",
      "Microsoft.Hosting.Lifetime": "Error"
    }
  }

...
```

CloudWatch

In case that application is hosted on AWS platform, CloudWatch logging can be enabled. Following example adds required part.

```
...

  "Logging": {
    "LogLevel": {
      "Default": "Error",
      "Microsoft": "Error",
      "Microsoft.Hosting.Lifetime": "Error"
    },
    "AWS":{
      "Enabled": true,
      "Region": "eu-central-1",
      "LogGroup": "/some-group",
      "LogLevel": {
        "Default": "Error",
        "Microsoft": "Error",
        "Microsoft.Hosting.Lifetime": "Error"
      }
    }
  }

...
```

Application Insights

Logging to AI is also supported. Two parts are needed. One on the same level as **"Logging"** object with **"InstrumentationKey"** value. Second is required only if logging level of AI needs to be adjusted to other than default "Warning" value.

```
...

  "ApplicationInsights": {
    "InstrumentationKey": "************************************"
  },
  "Logging": {
    "LogLevel": {
      "Default": "Error",
      "Microsoft": "Error",
      "Microsoft.Hosting.Lifetime": "Error"
    },
    "ApplicationInsights":{
      "LogLevel": {
        "Default": "Error",
        "Microsoft": "Error",
        "Microsoft.Hosting.Lifetime": "Error"
      }
    }
  }

...
```

Full example :

```
{
  "ConnectionStrings": {
    "CollaborationHubDB": "Server=db,1433;Initial Catalog=IBV.Database;Persist Security Info=False;User ID=sa;Password=Welcome@1;Integrated Security=false;Connection Timeout=30;",
    "IdentityDB": "Server=db,1433;Initial Catalog=IBV.Database;Persist Security Info=False;User ID=sa;Password=Welcome@1;Integrated Security=false;Connection Timeout=30;",
    "ConfigurationDB": "Server=db,1433;Initial Catalog=IBV.Database;Persist Security Info=False;User ID=sa;Password=Welcome@1;Integrated Security=false;Connection Timeout=30;",
    "LicensingDB": "Server=db,1433;Initial Catalog=IBV.Database;Persist Security Info=False;User ID=sa;Password=Welcome@1;Integrated Security=false;Connection Timeout=30;",
    "MFT_DB": "Server=db,1433;Initial Catalog=IBV.Database;Persist Security Info=False;User ID=sa;Password=Welcome@1;Integrated Security=false;Connection Timeout=30;"
  },
  "Cors": {
    "AllowedOrigins": [
      "https://localhost:8443",
      "chrome-search://local-ntp"
    ]
  },
```

```
  "Logging": {
    "LogLevel": {
      "Default": "Error",
      "Microsoft": "Error",
      "Microsoft.Hosting.Lifetime": "Error"
    }
  }
}
```

[[*TOC*]]

1. Frontend logging

Logging of the frontend application can be configured in **env.Override.js** file on the cb_frontend container. If any log provider is configured, logs will be sent according to the configured log level (optimal log level for production environments is warning or error). Additional SignalR or WebRTC log source can be enabled.

1.1. SignalR logging

SignalR communication is very frequent and thus its logging is not enabled by default. There are two properties of env.Override.js file that can be changed to enable it.

- **signalRLogging** -- when set to true, SignalR communication will be logged to the browser console
- **logSignalRMessages** -- when set to true, SignalR communication will be logged to the configured logging target.

```
window.runtimeConfig = {
  ...
  signalRLogging: true,
  logSignalRMessages: true,
  ...
}
```

1.2. Logging targets

Application Insights

To enable logging to application insights only key needs to be configured.

```
window.runtimeConfig = {
  ...
  applicationInsightsKey: "< APP INSIGHTS KEY >",
  ...
}
```

AWS CloudWatch

```
window.runtimeConfig = {
  ...
  isOnAws: true,
  aws: {
    accessKeyId: "< ACCESS KEY ID >",
    accessKey: "< ACCESS KEY >",
    region: "< REGION >",
    errorLogGroup: "Collaboard_Errors",
    eventLogGroup: "Collaboard_Events"
  },
  ...
}
```

2. Backend logging

In Collaboard backend we have a logging system and additional custom traces.

Logging of backend is configured in **appsettings.Override.json** in section **Logging**. LogLevel, which is configured on root of Logging object, filters messages entering logging pipeline. LogLevel in respective logging provider filters messages that are being picked from the logging pipeline.

In following example, default logging level for all providers is set to Information while Console logging provider is set to Error only.

```
{
  ...
  "Logging": {
    "LogLevel": {
      "Default": "Information"
    },
    "Console": {
      "LogLevel": {
        "Default": "Error"
      }
    },
    ...
  }
}
```

2.1. LogLevel

In general, LogLevel section contains key-value pairs where value is the logging level and key is the scope for which the logging level is being configured. Accepted LogLevel values are listed here.

Scopes

There are multiple logging scopes that can be configured:

- IBV.Telemetry -- this scope applies to all logs originating from Collaboard source code.
- IBV.Telemetry.UserAction -- specialized scope containing user actions only. Set to **Trace** by default.
- IBV.Telemetry.Anytime -- specialized scope that contains important informational traces. Set to **Trace** by default.
- IBV.Telemetry.Performance -- specialized scope for performance traces. Set to **None** by default.

Trace User Action

Trace user action is enabled by setting **LogLevel** for *IBV.Telemetry.UserAction* to **Trace**.

```
{
  ...
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "IBV.Telemetry.UserAction": "Trace"
    },
    ...
  }
}
```

Traces use the "USER ACTION" prefix. This tracing is **enabled** by default.

Traces can be found in the **cb-auth**, **cb-api** and **mft-web** container logs.

It traces:

- Login
- Logout
- Failed login
- Board/project open
- User starts upload (UploadStarted in the picture above) (MFT only)
- User finished upload (UploadCompleted in the image above) (MFT only)
- Access denied for an upload request (MFT only)
- Stream (stream download via range headers) (MFT only)
- User start upload (MFT only)
- User finish download (MFT only)
- Access denied for a download request (MFT only)

In the sample below you can notice a Board/project being open by user gianpaolo.santopaolo@gmail.com for project id 112 and the date when the event happened.

```
2021-04-07T15:28:17.832+02:00          USER ACTION - GetProject - User: gianpaolo.santopaolo@gmail.com - Project: 112
```

Performance traces

Performance traces are enabled by setting **LogLevel** for *IBV.Telemetry.Performance* to **Trace**.

```
{
  ...
  "Logging": {
```

```
    "LogLevel": {
      "Default": "Information",
      "IBV.Telemetry.Performance": "Trace"
    },
    ...
  }
}
```

Traces use the "PERF" prefix. This tracing is **disabled** by default.

Anytime logs

Performance traces are enabled by setting **LogLevel** for *IBV.Telemetry.Anytime* to **Trace**.

```
{
  ...
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "IBV.Telemetry.AnyTime": "Trace"
    },
    ...
  }
}
```

Traces use the "ANYTIME" prefix. This tracing is **enabled** by default.

Additional useful scopes

- Default -- this scope is being applied to all scopes that have no scope specified.
- Microsoft.AspNetCore.SignalR -- this scope can be used to debug SignalR communication.
- Microsoft.AspNetCore.Http.Connections -- this scope can be used to debug connections, i.e. WebSockets.

2.2. Logging providers

Multiple logging providers can be configured. They are not mutually exclusive and can be used side by side.

All following providers may contain LogLevel object to set different logging levels for each one of them.

Console

Logging to console is configured/enabled using the "Console" object in "Logging".

```
{
  ...
  "Logging": {
    ...
    "Console": {
      "LogLevel": {
        "Default": "Warning"
      }
    },
    ...
  }
}
```

Application insights

To enable logging to Application Insights, there are two places in appsettings.Override.json .

- On root level you need to specify Application Insights key.
- In "Logging" object you can define "ApplicationInsights" object with "LogLevel" to override the default setting.

```
{
  "ApplicationInsights": {
    "InstrumentationKey": "<key>"
  },
  ...
  "Logging": {
    ...
    "ApplicationInsights": {
      "LogLevel": {
        "Default": "Warning"
      }
    },
    ...
  }
}
```

CloudWatch

AWS Cloud exclusive only. CloudWatch logging has to be enabled using the **Logging → AWS → Enabled** property. Library **AWS.Logger.AspNetCore** is used, additional information can be found here.

```
{
  ...
  "Logging": {
    ...
    "AWS": {
      "Enabled": "true",
      "Region": "eu-central-1",
      "LogGroup": "IBV.CL.API",
      "IncludeLogLevel": true,
      "IncludeCategory": true,
      "IncludeNewline": true,
      "IncludeException": true,
      "IncludeEventId": false,
      "IncludeScopes": false,
      "LogLevel": {
        "Default": "Error"
      }
    },
    ...
  }
}
```

3. Full example

In following example all three providers are configured:

- **Console** -- logging Warning and all three special scopes.
- **Application Insights** -- logging Errors and UserAction + Anytime traces.
- **CloudWatch** -- logging Errors only.

```
{
  ...
  "ApplicationInsights": {
    "InstrumentationKey": "<key>"
  },
  "Logging": {
    "LogLevel": {
      "Default": "Warning",
      "IBV.Telemetry.UserAction": "Trace",
      "IBV.Telemetry.Anytime": "Trace",
      "IBV.Telemetry.Performance": "Trace"
    },
    "Console": {
      "LogLevel": {
        "Default": "Warning",
        "IBV.Telemetry.UserAction": "Trace",
        "IBV.Telemetry.Anytime": "Trace",
        "IBV.Telemetry.Performance": "Trace"
      }
```

```
    },
    "ApplicationInsights": {
      "LogLevel": {
        "Default": "Error",
        "IBV.Telemetry.UserAction": "Trace",
        "IBV.Telemetry.Anytime": "Trace"
      }
    },
    "AWS": {
      "Enabled": true,
      "Region": "eu-central-1",
      "LogGroup": "IBV.CL.API",
      "LogLevel": {
        "Default": "Error"
      }
    }
  }
}
```

Scope

The purpose of this page is to explain how to configure your Reserve Proxy in order to make it work. As there are many Reverse Proxies on the market, we can not list all possible configurations.

Reverse Proxies we have tested:

1. NGINX reserve proxy
2. Apache Reserve proxy on Debian based Linux systems (e.g. like Ubuntu)

Note: In case of any other reverse proxy configurations, we can not guarantee that it will work. It usually requires to find the right options/settings.

Examples configurations:

NGINX Reserve Proxy :

```
user nginx;
worker_processes auto;
error_log /var/log/nginx/error.log;
pid /run/nginx.pid;
include /usr/share/nginx/modules/*.conf;

events {
    worker_connections 1024;
}

http {
    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';

    access_log  /var/log/nginx/access.log  main;

    sendfile            on;
    tcp_nopush          on;
    tcp_nodelay         on;
    keepalive_timeout   65;
    types_hash_max_size 4096;

    include             /etc/nginx/mime.types;
    default_type        application/octet-stream;

    include /etc/nginx/conf.d/*.conf;

    server {
        listen       443 ssl http2;
        listen       [::]:443 ssl http2;
        server_name  _;
        root         /usr/share/nginx/html;

        ssl_certificate "/etc/ssl/certs/cert.pem";
        ssl_certificate_key "/etc/ssl/certs/privkey.pem";
        ssl_session_cache shared:SSL:1m;
        ssl_session_timeout  10m;
        ssl_ciphers PROFILE=SYSTEM;
        ssl_prefer_server_ciphers on;

        location /server/signalr {
            proxy_pass         wss://internal.server.com:8443/server/signalr;
            proxy_http_version 1.1;
            proxy_set_header   Upgrade $http_upgrade;
            proxy_set_header   Connection upgrade;
            proxy_set_header   Host $host;
            proxy_cache_bypass $http_upgrade;
            proxy_set_header   X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header   X-Forwarded-Proto $scheme;
        }

        location / {
            proxy_pass         https://internal.server.com:8443/;
            proxy_redirect     off;
            proxy_set_header   Host $host;
            proxy_set_header   X-Real-IP $remote_addr;
            proxy_set_header   X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header   X-Forwarded-Host $server_name;
        }
        # Load configuration files for the default server block.
        include /etc/nginx/default.d/*.conf;

        error_page 404 /404.html;
            location = /40x.html {
        }

        error_page 500 502 503 504 /50x.html;
            location = /50x.html {
        }
    }
}
```

Apache Reverse Proxy on Ubuntu :

```
<VirtualHost *:443>
  ServerName external.companyx-public-ok.net

  ## Logging
  ErrorLog "/var/log/httpd/external.companyx-public-ok.net-443_error_ssl.log"
  ServerSignature Off
  CustomLog "/var/log/httpd/external.companyx-public-ok.net-443_access_ssl.log" combined

  ## Header rules
  Header always set Strict-Transport-Security "max-age=31536000"
  ProxyPreserveHost On

  ## SSL directives
  SSLEngine on
  SSLCertificateFile      "/etc/ssl/companyx-public-ok-net/wildcard/companyx-public-ok-net.crt"
  SSLCertificateKeyFile   "/etc/ssl/companyx-public-ok-net/wildcard/companyx-public-ok-net.key"
  SSLCertificateChainFile "/etc/ssl/companyx-public-ok-net/wildcard/companyx-public-ok-net.bundle.crt"

  ## Custom fragment

            RewriteCond %{HTTP_HOST} !^(collaboard-internal\.live\.companyx\.org|external\.companyx-public-ok\.net)$
            RewriteRule ^(.*)$ https://external.companyx-public-ok.net [R=301,NC]
            ProxyPass /server/signalr wss://localhost:8443/server/signalr
            ProxyPass   /      https://localhost:8443/
              # SSL options
            SSLProxyEngine          On
            SSLProxyVerify          none
            SSLProxyCheckPeerCN     off
            SSLProxyCheckPeerName   off
            SSLProxyCheckPeerExpire off

</VirtualHost>
```

Notes

What is important in these reverse proxy configurations ?

- It must be using websockets for all URI paths /server/signalr
- All other communications should use https
- Should have the proper certificates installed
- websocket libraries must be installed and loaded

[[*TOC*]]

Scope

This is an optional step and is only needed in case a container (e.g. cb-auth) needs to have a secure SSL connection between your internal system (e.g. mail server) and Collaboard.

Importing internal CA certificate

In Docker-compose

Importing CA cert needs to be done on Operating system level. Docker simply relies on CAs installed on underlying operating system.

In case you do not have access to the underlying operating system for any reason, you can add the TLS certificate into the container as documented below.

Rebuild the docker image doing the follwing commands :

Dockerfile:

```
FROM cb-auth:<TAG>
ADD your_ca_root.crt /usr/local/share/ca-certificates/my-cert.pem
RUN chmod 644 /usr/local/share/ca-certificates/my-cert.pem && update-ca-certificates
```

Build:

```
docker build -t <build-name> -f Dockerfile .
```

In order to use this new image, you will need to update your docker-compose.yml file and restart the application.

In Kubernetes or OpenShift

Importing CA cert needs to be done on every cluster node: all masters and workers. Kubernetes simply relies on CAs installed on underlying operating system.

In case you do not have access to the underlying operating system, you can add the TLS certificate into the POD or container's trusted root certificate store as documented below.

1. Create config map using .pem file

```
kubectl -n <namespace-for-config-map-optional> create configmap ca-pemstore —from-file=my-cert.pem
```

2. Mount that configmap into the container

's file as one to one file relationship in volume mount in directory /etc/ssl/certs/ as file for example :

```
apiVersion: v1
kind: Pod
metadata:
  name: cb-auth
spec:
      containers:
      - name: cb-auth
        image: cacheconnectsample:v1
        volumeMounts:
        - name: ca-pemstore
          mountPath: /etc/ssl/certs/my-cert.pem
          subPath: my-cert.pem
          readOnly: false
      volumes:
      - name: ca-pemstore
        configMap:
          name: ca-pemstore
```

**Note ⚠* This manifest is not complete, you need to merge your existing deployment with the additional configmap.

[[*TOC*]]

Scope

This guide describes the installation of Collaboard on Azure AKS. Azure AKS is a technology that you can use to run containers without having to manage a Kubernetes Control Plane. AKS is an open-source fully managed container orchestration service and is available on the Microsoft Azure public cloud that can be used to deploy, scale and manage Docker containers and container-based applications in a cluster environment. Azure Kubernetes Service offers provisioning, scaling, and upgrades of resources as per requirement or demand without any downtime in the Kubernetes cluster.

Architecture overview

Each Collaboard container is running in a corresponding Kubernetes POD. The AKS cluster will be deployed in one resource group in a private subnet. Only the cb-proxy container will have access to the "outside" through a loadbalancer. The frontend containers (frontend, auth, api, ...) need to communicate with through a separated loadbalancer which is completely private and internal.

Infrastructure deployment of the AKS Cluster

The deployment of the AKS Cluster is done through simple CLI commands.

The following snippet is an example of an AKS cluster deployment and can be adapted for your environment :

```
MYIP="80.1.1.10"
TSHIRTSIZE="Standard_F8s_v2"
NR_OF_NODES=2
MAX_NR_OF_NODES=3
GROUP_SEQ=13
AKS_SEQ=01

regionx () {
  az login
  az group create --name AZ99A000PRSG00${GROUP_SEQ}  --location uswest1
  az aks create \
    --resource-group AZ99A000PRSG00${GROUP_SEQ} \
    --name AZ99A000PAKS00${AKS_SEQ} \
    --node-count $NR_OF_NODES \
    --nodepool-name az99wenodep1 \
    --vm-set-type VirtualMachineScaleSets \
    --load-balancer-sku standard \
    --api-server-authorized-ip-ranges $MYIP/32 \
    --node-vm-size $TSHIRTSIZE \
    --enable-cluster-autoscaler \
    --min-count $NR_OF_NODES \
    --max-count $MAX_NR_OF_NODES \
    --generate-ssh-keys
  az aks get-credentials --resource-group AZ99A000PRSG00${GROUP_SEQ} --name AZ99A000PAKS00${AKS_SEQ}
  kubectl get nodes
}
regionx

exit 0
```

Collaboard application Installation and configuration

You need to follow the instructions "here"

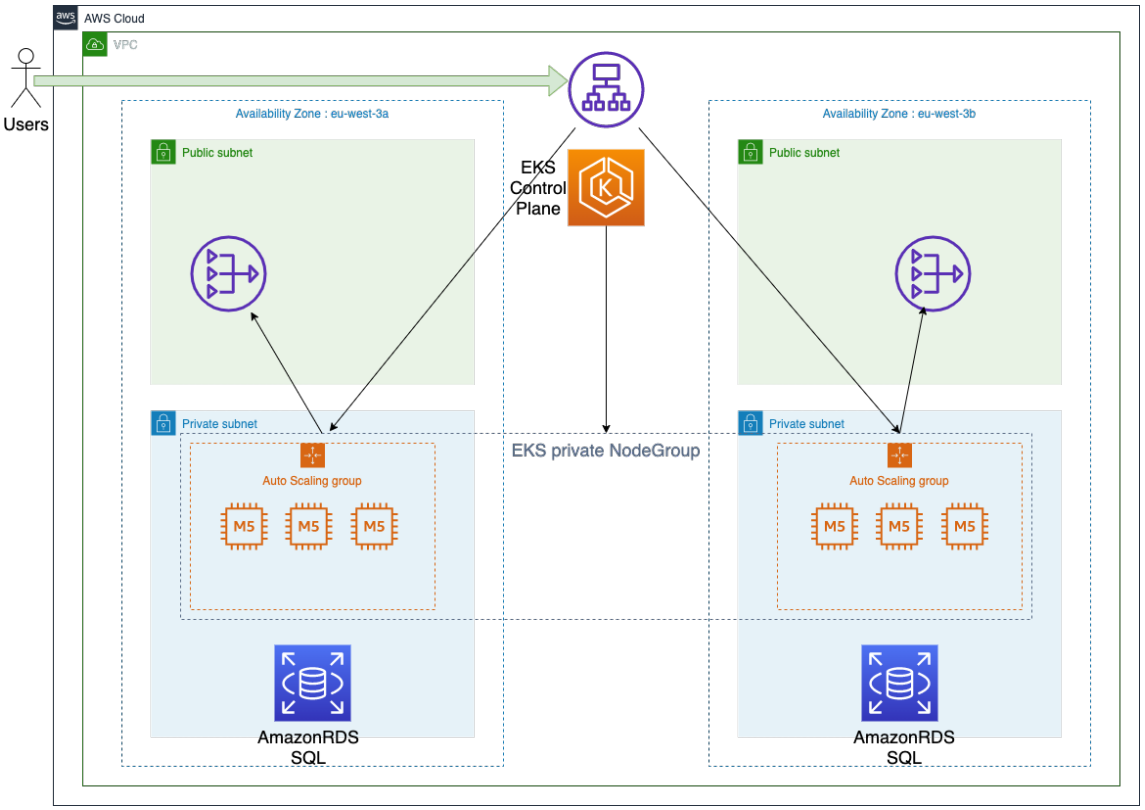
[[*TOC*]]

Scope

This guide describes the installation of Collaboard on AWS EKS. AWS EKS is a technology that you can use to run containers without having to manage a Kubernetes Control Plane. EKS is an open-source fully managed container orchestration service and is available on the AWS public cloud that can be used to deploy, scale and manage Docker containers and container-based applications in a cluster environment. AWS Kubernetes Service offers provisioning, scaling, and upgrades of resources as per requirement or demand without any downtime in the Kubernetes cluster.

Architecture overview

Each Collaboard container is running in a corresponding Kubernetes POD. The EKS cluster will be deployed in one resource group in a private subnet. Only the cb-proxy container will have access to the "outside" through a loadbalancer. The frontend containers (frontend, auth, api, ...) need to communicate with through a separated loadbalancer which is completely private and internal.



Infrastructure deployment of the EKS Cluster

The deployment of the EKS Cluster is done through simple AWS CLI commands.

High level instructions

Step	Action	Action Type	Duration
1	Deploy AWS VPC network infrastructure	script	10m
2	Deploy AWS EKS infrastructure	script	15m
3	Provision AWS RDS MSSQL instance	Manual (GUI)	15m
4	Provision AWS ElastiCache Redis instance	Manual (GUI)	15m
5	Create AWS EFS shared file system	Manual (CLI)	10m
6	Set the kubernetes cluster admin credential	Manual (CLI)	1m
7	Deploy EFS CSI driver for kubernetes	script	15m
8	Deploy application containers/manifests	script	15m
9	Update DNS	Manual	10m

Detailed installation instructions Collaboard Installation On AWS EKS

Collaboard application Installation and configuration

You need to follow the instructions "here"

[[*TOC*]]

Quickstart

1. Install all prerequisites.
2. Ensure you have required access to resources.
3. Start Docker Desktop.
4. Git clone the Distribution repository
   - HTTPS: `git clone https://ibvsolutions@dev.azure.com/ibvsolutions/CollaboardServices/_git/Distribution`
   - SSH: `git clone git@ssh.dev.azure.com:v3/ibvsolutions/CollaboardServices/Distribution`
5. Adjust variables in .env file.
6. Prepare appsettings.Override.json file if necessary.
7. Run **update-and-run.sh** (Linux) or **update-and-run.ps1** (Windows) script.
   - If you have yet to log in for the first time, it will ask for credentials.
   - You will be asked whether to update DB. The safe choice is yes (just hit enter) unless you know you don't need it.
8. Make sure all containers are up and running (docker container ls -a)
9. Access the website on https://localhost:8443/
   - You might get error code 400 if you try using HTTP. In that case, refresh the page, and it will redirect to HTTPS.
   - When you browse the page, if you haven't installed the self-signed certificate on your machine, you might be informed that the site is not secure. You shall proceed

Installation

There are two ways to install the containers. One is manual, second is by using one of the prepared scripts. For both .env file is used:

.env file

The .env file contains a set of variables that can be changed before execution. Most of them are, in a way, a variable or mount from Configuration wiki.

- **ENV_REGISTRY** -- defines source of containers, production registry is ibvcollaboard.azurecr.io.
- **ENV_VERSION** -- basically defines which version should be downloaded, default is **latest**, numbered version is advised (i.e. 4.5.4542).
- **SQL_DATA** -- the path to the folder where SQL persists database data.
- **SQL_LOGS** -- the path to the folder where SQL persists database logs.
- **SQL_SCRIPTS** -- the path to the folder where the SQL configuration script is. It is required in case the script filename is defined.
- **SQL_SCRIPT_FILE** -- filename of SQL configuration script.
- **SA_PASSWORD** -- password for sa user of the database.
- **SSL_CERTS** -- path to folder with certificate files (*.crt + *.key).
- **CB_FILES** -- the path to the folder where application persists project files.
- **CB_TEMP** -- the path to the folder where the application stores temporary data.
- **CB_OVERRIDES** -- the path to the folder with appsettings override file.
- **CB_OVERRIDE_FILE** -- filename of appsettings override file.
- **CB_OVERRIDE_ENV_JS** -- filename of frontend env.js override file.
- **CB_OVERRIDE_NGINX** -- filename of proxy nginx.conf override file.
- **CB_OVERRIDE_TURN** -- filename of turnserver.conf file.
- **TURN_REALM** -- TURN server realm.
- **TURN_SERVER** -- TURN server realm.
- **TURN_MIN_PORT** -- TURN server realm.
- **TURN_MAX_PORT** -- TURN server realm.
- **TURN_USER** -- TURN server realm.
- **TURN_PSWD** -- TURN server realm.

!! IMPORTANT !! If you change the ENV_VERSION after having it already installed, it will apply the scripts with target version. Switching to older versions might be impossible with some breaking changes, for example removal of project key in paths.

More information about override files and script can be found in respective sub-pages under Configuration wiki.

Note: for the variables pointing to a path, use the appropriate Operating System standards. Windows is using drive letters while Linux is using forward slashes. example for Windows : c:/*SQL/DATA* example for Linux : /sql/data

1. Script

In order to have the system running/updated as easily as possible, there are two scripts prepared. Both contain the same steps, it depends on the operating system which one it can use.

- **update-and-run.ps1** (on Windows)
- **update-and-run.sh** (on Linux)

Scripts require the following files/folders to be present in the same directory:

- **.env** – contains adjustable variables
- **cb-proxy.crt**, **cb-proxy.key** – their location can be changed in .env file, they represent SSL certificate used for https communication
- **cb-coturn.pem**, **cb-coturn.key** – their location can be changed in .env file, they represent SSL certificate used for https communication of TURN server
- **ConfigurationMigration.sql** – it's location can be changed in .env file, it contains configuration values to be applied
- **license.lic** – it needs to be mounted to all containers, it's location inside the container can be changed in ConfigurationMigration.sql
- **docker-compose.yml** – compose file declaring the system structure and configuration

First check the .env file values, whether you want to change some paths, version, etc. If you want to change the path where the SQL server stores the database data to for example /usr/tmp/sql_data, then you need to change the line

`SQL_DATA=c:/_SQL_/_DATA_` to `SQL_DATA=/user/tmp/sql_data`

After script execution, there might be a prompt for login. After login, there will be a prompt asking whether to update the database as well. Always update the database unless you know there were no changes. But if skipped, it saves a lot of time.

Re-running the script will update to the latest version, if available and "latest" tag is used. The database is always updated unless skipped.

2. Certificate generation

For Windows

There is a PowerShell script in the folder called certs, with the name **generate-localhost-certs.ps1** which will generate the localhost SSL certificates **cb-proxy.crt, cb-proxy.key**

For Linux

```
openssl req -x509 -nodes -days 365 -subj "/C=CH/O=IBV/CN=localhost" -newkey rsa:2048 -keyout cb-proxy.key -out cb-proxy.crt
```

Please note that the -subj parameters needs to be adapted according to your environment C : Country O : Organisation

3. Manual

Following are basically manual steps that are part of the previous scripts. Replace the C:_SQL_ and similar values with their equivalent from .env file. Also, adjust the .env file accordingly before doing these steps.

The --env-file=.env parts can be removed, .env is included by default but it might come in handy in case of multiple files for various envs.

3.1. Install/Update containers

When you have the main docker-compose.yml file, all you need to do to get a up and running system is issue the command:

```
docker-compose -f "docker-compose.yml" --env-file=$ENV_FILE up --no-start
docker start cb_db.$ENV_VERSION
```

This will initially draw all images from the repository, this might take a long time! On first installation, only cb_db container will be up and running. In case of update, all containers will be updated and kept running during the next steps.

There needs to be SSL certificate pair present on the relative path to the docker-compose.yml. The path can be changed in the .env file.

- certs\localhost\cb-proxy.crt
- certs\localhost\cb-proxy.key

Subsequent updates can be done using these commands:

PowerShell

```
docker run -it --name cb_database_init `
               --env-file=$ENV_FILE `
               --network=cbnetwork `
               -v ${SQL_SCRIPTS}:/var/opt/mssql/migration_script `
               ibvcollaboard.azurecr.io/cb_database_init:latest

docker rm cb_database_init
docker-compose -f "docker-compose.yml" --env-file=.env pull
docker-compose -f "docker-compose.yml" --env-file=.env up -d
docker-compose -f "docker-compose.yml" --env-file=.env restart
```

Bash

```
docker run -it --name cb_database_init \
               --env-file=$ENV_FILE \
               --network=cbnetwork \
               -v ${SQL_SCRIPTS}:/var/opt/mssql/migration_script \
               ibvcollaboard.azurecr.io/cb_database_init:latest

docker rm cb_database_init
docker-compose -f "docker-compose.yml" --env-file=.env pull
docker-compose -f "docker-compose.yml" --env-file=.env up -d
docker-compose -f "docker-compose.yml" --env-file=.env restart
```

3.2. Prepare database

The data in the database server is stored in the share C:_SQL_. So this folder should contain all necessary files for SQL to work properly. To get those files, one need to run the following command once while cb_db container is up and running with:

PowerShell

```
docker run -it --name cb_database_init `
               --env-file=$ENV_FILE `
               --network=cbnetwork `
               -v ${SQL_SCRIPTS}:/var/opt/mssql/migration_script `
               ibvcollaboard.azurecr.io/cb_database_init:latest

docker rm cb_database_init
```

Bash

```
docker run -it --name cb_database_init \
               --env-file=$ENV_FILE \
               --network=cbnetwork \
               -v ${SQL_SCRIPTS}:/var/opt/mssql/migration_script \
               ibvcollaboard.azurecr.io/cb_database_init:latest

docker rm cb_database_init
```

This will pull the correct image and run the container. Inside the container are all the commands needed to create/update the database. After this is done, the system will be ready and we don't need this container anymore.

3.3. Restart containers

Final step is to restart all containers to make sure all are running with up to date cached from database.

```
docker-compose -f "docker-compose.yml" restart
```

Once this is done, the system will start and Collaboard can be used by going to a browser and opening the page https://localhost:8443

[[*TOC*]]

Installation of Collaboard on K8s

1. Install all CLI utilities from the "Pre-requisites" section.
2. Ensure you have all the required accesses to the resources as documented in the "Pre-requisites" section.
3. Prepare the necessary configuration files :
   a. **appsettings.Override.json** (CB_OVERRIDE_FILE) : Consult the Pre-requisites and Configuration section in order to preparing this file. Prepare your specific configurations and save them in the file `appsettings.Override.json`.
   b. **env.Override.js** (CB_OVERRIDE_ENV_JS) : Consult the Pre-requisites and Configuration section in order to preparing this file. Prepare your specific configurations and save them in the file `env.Override.js`.
   c. **backoffice.Override.js** (CB_OVERRIDE_BACKOFFICE) : Consult the Pre-requisites and Configuration section in order to preparing this file. Prepare your specific configurations and save them in the file `backoffice.Override.js`.
   d. **ConfigurationMigration.sql** (SQL_SCRIPT_FILE) : Consult the Pre-requisites and Configuration section in order to preparing this file. Prepare your specific configurations and save them in the file `ConfigurationMigration.sql`.
   e. **envoy.Override.yaml** (CB_OVERRIDE_ENVOY) : Consult the Pre-requisites and Configuration section in order to preparing this file. Prepare your specific configurations and save them in the file `envoy.Override.yaml`.
4. Git clone the Distribution repository

- for HTTPS:

```
git clone https://ibvsolutions@dev.azure.com/ibvsolutions/CollaboardServices/_git/Distribution
```

- for SSH:

```
git clone git@ssh.dev.azure.com:v3/ibvsolutions/CollaboardServices/Distribution
```

5. Create a new namespace (this is optional)

```
kubectl create ns collaboard
```

6. Change namespace context (only if you have created a new namespace)

```
kubectl config set-context --current --namespace=collaboard
```

7. Create the required persistent volumes

```
cd Distribution/Kubernetes
kubectl create -f ./persistentvolumeclaims/
```

To check: `kubectl get pvc` In the output you should see 4 volumes in the STATUS **"BOUND" Note**: In case of any problem with the creation of the persistent volumes, it might be needed to modify the manifests in order to match the storage class that you have configured in Kubernetes.

8. Create the required configmaps

```
kubectl create cm cm-appsettings --from-file=appsettings.Override.json
kubectl create cm cm-env-settings --from-file=env.Override.js
kubectl create cm cm-backoffice-settings --from-file=backoffice.Override.js
kubectl create cm sql-init-script --from-file=ConfigurationMigration.sql
kubectl create cm cm-envoyconf --from-file=envoy.Override.yaml
kubectl create -f ./configmaps/
```

- To check: `kubectl get cm` In the output, you should see at least 6 configmaps

```
NAME                     DATA   AGE
cm-appsettings           1      12m
cm-env-settings          1      12m
cm-backoffice-settings   1      12m
cm-envoyconf             1      12m
cm-nginx-frontend        1      12m
sql-init-script          1      12m
```

9. Create the required secrets

```
kubectl create -f ./secrets/
```

- To check:

```
kubectl get secrets
```

- In the output, you should see the following 2 additional secrets that have been created.

```
NAME                  TYPE                             DATA   AGE
db-secret             Opaque                           2      7s
pull-secret           kubernetes.io/dockerconfigjson   1      8s
```

Note: In case you have updated the appsettings.Override.json with a different DB password, you need also to update the db-secret with the new password.

10. Start the Database container (in case you run it in a container)

```
kubectl create -f ./deployments/cb-db.yaml
```

- To check:

```
kubectl get po | grep db
```

- In the output, you should see the POD that have been created and in running state. If not, check the logs with `kubectl logs <POD-NAME>` example : `kubectl logs db-xyzaaa`

11. Run the DB initialization script via a Kubernetes Job

```
kubectl create -f ./jobs/
```

- To check:

```
kubectl get po,job | grep db-init
```

- In the output, you should see the job and POD that have been created. To verify a successful completion: `kubectl get job` In the output, you should see the job with **COMPLETIONS 1/1** If not successful, check the logs with `kubectl logs <POD-NAME>` example : `kubectl logs db-init-lqjlq`

```
NAME                READY   STATUS      RESTARTS   AGE
pod/db-init-lqjlq   0/1     Completed   0          10m

NAME                COMPLETIONS   DURATION   AGE
job.batch/db-init   1/1           3m33s      10m
```

12. Deploy all other containers

```
kubectl create -f ./deployments/
```

- To check:

```
kubectl get po,deployment
```

- In the output you should see the containers being created :

```
NAME                                          RDY  STATUS              RESTS  AGE
pod/cb-api-6b87544699-gft9b                   0/1  ContainerCreating   0      10s
pod/mft-chunkservice-5867967447-8c476         0/1  Pending             0      8s
pod/mft-cleanerservice-5df95499cd-zrvw7       0/1  Pending             0      8s
pod/mft-mergeservice-66d94d576d-clwxm         0/1  Pending             0      8s
pod/mft-storageoperationsservice-6dd6449-xgzdr 0/1 Pending             0      8s
pod/cb-copyworker-79d88d8d86-62vtg            0/1  ContainerCreating   0      10s
pod/cb-db-67fd599984-pfkr2                    0/1  ContainerCreating   0      10s
pod/cb-fileconverter-cf5577c49-vkxz4          0/1  ContainerCreating   0      10s
pod/cb-auth-687df8fd-mmw2r                    0/1  ContainerCreating   0      10s
pod/cb-canvasshotter-68f874c4dd-9d5mw         0/1  ContainerCreating   0      10s
pod/cb-frontend-78578cfb7f-sxrfp              0/1  ContainerCreating   0      9s
pod/cb-licensing-6cd77bd4d-zk8t4              0/1  ContainerCreating   0      9s
pod/cb-proxy-5b5b74fd87-5n4z5                 0/1  ContainerCreating   0      9s
pod/mft-web-6d4fccfc65-2tqjv                  0/1  Pending             0      7s
pod/cb-worker1-f848775d4-hpjsf                0/1  ContainerCreating   0      9s
pod/cb-worker2-79797b87c7-p7k4r               0/1  ContainerCreating   0      8s
```

Once completed, all POD should be in **Running** STATUS

13. Create the services for the different POD

```
kubectl create -f ./services/
```

- To check:

```
kubectl get svc
```

- In the output, you should see all the services that have been created.

14. Create Ingress for routing the traffic into the application. This step will be specific to your environment depending on which kind of ingress-controller has been installed. The following step is just an example. Update the provided example manifest accordingly. Update the manifest called `ingress/ingress-cb-proxy.yaml` with the URL to access the collaboard application (e.g. collaboard.example.com) Save and create the manifest with :

```
kubectl create -f ./ingress/
```

- To check:

```
kubectl get ingress
```

- In the output, you should see all the ingress routes that has been created.

15. Access website on https://

- You might get error code 400 if you try using HTTP. In that case, just refresh the page and it will redirect to HTTPS. **Important Note:** In case you want to run the cb_proxy container on a different port, you will need to adapt the proxy service referencing the container port. Besides updating the service, you need to update as well the cm-envoyconf configmap accordingly as it is listening on port 8443 by default.

16. Install SSL certificate Take your SSL certificate (<your-SSL.cert>.crt) and private key (<your-SSL.cert>.key) matching your URL and generate a Kubernetes secret from it. Rename the .crt and .key file to cb-proxy.crt and cb-proxy.key and execute : `kubectl create secret generic cb-proxy-ssl --from-file=cb-proxy.key --from-file=cb-proxy.crt` and restart the cb-proxy POD : `kubectl delete po cb-proxy-xxxx` (with xxxx the name of the POD)

17. Installation of your license Consult the section "Activating-your-License"

18. In case Kubernetes is behind a reverse proxy Consult the Configuration section "Collaboard-behind-a-reverse-proxy"

[[*TOC*]]

Installation of Collaboard on OpenShift

1. Install all CLI utilities from the "Pre-requisites" section.
2. Ensure you have all the required accesses to the resources as documented in the "Pre-requisites" section.
3. Prepare the necessary configuration files :
   a. **appsettings.Override.json** (CB_OVERRIDE_FILE) : Consult the Pre-requisites and Configuration section in order to preparing this file. Prepare your specific configurations and save them in the file `appsettings.Override.json`.
   b. **env.Override.js** (CB_OVERRIDE_ENV_JS) : Consult the Pre-requisites and Configuration section in order to preparing this file. Prepare your specific configurations and save them in the file `env.Override.js`.
   c. **backoffice.Override.js** (CB_OVERRIDE_BACKOFFICE) : Consult the Pre-requisites and Configuration section in order to preparing this file. Prepare your specific configurations and save them in the file `backoffice.Override.js`.
   d. **ConfigurationMigration.sql** (SQL_SCRIPT_FILE) : Consult the Pre-requisites and Configuration section in order to preparing this file. Prepare your specific configurations and save them in the file `ConfigurationMigration.sql`.
   e. **envoy.Override.yaml** (CB_OVERRIDE_ENVOY) : Consult the Pre-requisites and Configuration section in order to preparing this file. Prepare your specific configurations and save them in the file `envoy.Override.yaml`.
4. Git clone the Distribution repository
   - HTTPS: `git clone https://ibvsolutions@dev.azure.com/ibvsolutions/CollaboardServices/_git/Distribution`
   - SSH: `git clone git@ssh.dev.azure.com:v3/ibvsolutions/CollaboardServices/Distribution`
5. Create a new OpenShift project/namespace

```
oc new-project collaboard
```

6. Create the required persistent volumes

```
cd Distribution/Kubernetes
oc create -f ./persistentvolumeclaims/
```

- To check: `oc get pvc`
- In the output, you should see 4 volumes in the STATUS **"BOUND" Note**: In case of any problem with the creation of the persistent volumes, it might be needed to modify the manifests in order to match the storage class that you have configured in OpenShift.

7. Create the required configmaps

```
oc create cm cm-appsettings --from-file=appsettings.Override.json
oc create cm cm-env-settings --from-file=env.Override.js
oc create cm cm-backoffice-settings --from-file=backoffice.Override.js
oc create cm sql-init-script --from-file=ConfigurationMigration.sql
oc create cm cm-envoyconf --from-file=envoy.Override.yaml
```

- To check: `oc get cm`
- In the output you should see 5 configmaps

```
NAME                     DATA   AGE
cm-appsettings           1      12m
cm-env-settings          1      12m
cm-backoffice-settings   1      12m
cm-envoyconf             1      12m
sql-init-script          1      12m
```

8. Create the required secrets

```
oc create -f ./secrets/
```

- To check:

```
oc get secrets
```

- In the output, you should see 2 additional secrets that have been created.

```
NAME                  TYPE                             DATA   AGE
pull-secret           kubernetes.io/dockerconfigjson   1      8s
db-secret             Opaque                           2      7s
```

Note: In case you have updated the appsettings.Override.json with a different DB password, you need also to update the db-secret with the new password.

9. Start the Database container (in case you run it in a container)

```
oc create -f ./deployments/cb-db.yaml
```

- To check:

```
oc get po | grep db
```

- In the output, you should see the POD that have been created and in running state. If not, check the logs with `oc logs <POD-NAME>` example : `oc logs db-xyzaaa`

10. Run the DB initialization script via an OpenShift Job

```
oc create -f ./jobs/
```

- To check: `oc get po,job | grep db-init`
- In the output, you should see the job and POD that have been created. To verify a successful completion: `oc get job` In the output, you should see the job with **COMPLETIONS 1/1** If not successful, check the logs with `oc logs <POD-NAME>` example : `oc logs db-init-lqjlq`

```
NAME                READY   STATUS      RESTARTS   AGE
pod/db-init-lqjlq   0/1     Completed   0          10m

NAME                COMPLETIONS   DURATION   AGE
job.batch/db-init   1/1           3m33s      10m
```

11. Deploy all other containers

```
oc create -f ./deployments/`
```

- To check: `oc get po,deployment`
- In the output you should see the containers being created.

```
NAME                                          RDY  STATUS              RESTS  AGE
pod/cb-api-6b87544699-gft9b                   0/1  ContainerCreating   0      10s
pod/mft-chunkservice-5867967447-8c476         0/1  Pending             0      8s
pod/mft-cleanerservice-5df95499cd-zrvw7       0/1  Pending             0      8s
pod/mft-mergeservice-66d94d576d-clwxm         0/1  Pending             0      8s
pod/mft-storageoperationsservice-6dd6449-xgzdr 0/1 Pending             0      8s
pod/cb-copyworker-79d88d8d86-62vtg            0/1  ContainerCreating   0      10s
pod/cb-db-67fd599984-pfkr2                    0/1  ContainerCreating   0      10s
pod/cb-fileconverter-cf5577c49-vkxz4          0/1  ContainerCreating   0      10s
pod/cb-auth-687df8fd-mmw2r                    0/1  ContainerCreating   0      10s
pod/cb-canvasshotter-68f874c4dd-9d5mw         0/1  ContainerCreating   0      10s
pod/cb-frontend-78578cfb7f-sxrfp              0/1  ContainerCreating   0      9s
pod/cb-licensing-6cd77bd4d-zk8t4              0/1  ContainerCreating   0      9s
pod/cb-proxy-5b5b74fd87-5n4z5                 0/1  ContainerCreating   0      9s
pod/mft-web-6d4fccfc65-2tqjv                  0/1  Pending             0      7s
pod/cb-worker1-f848775d4-hpjsf                0/1  ContainerCreating   0      9s
pod/cb-worker2-79797b87c7-p7k4r               0/1  ContainerCreating   0      8s
```

Once completed, all POD should be in **Running** STATUS

12. Create the services for the different POD

```
oc create -f ./services/
```

- To check: `oc get svc`
- In the output, you should see all the services that have been created.

13. Create an OpenShift route for routing the traffic into the application POD. Save and create the manifest with :

```
oc create route passthrough --service proxy --hostname <application-URL>
```

- To check: `oc get route`
- In the output, you should see the route that has been created.

14. Access website on https://

- You might get error code 400 if you try using HTTP. In that case, just refresh the page and it will redirect to HTTPS. **Important Note:** In case you want to run the cb_proxy container on a different port, you will need to adapt the proxy service referencing the container port. Besides updating the service, you need to update as well the cm-envoyconf configmap accordingly as it is listening on port 8443 by default.

15. Install SSL certificate Take your SSL certificate (<your-SSL.cert>.crt) and private key (<your-SSL.cert>.key) matching your URL and generate an OpenShift secret from it. Rename the .crt and .key file to cb-proxy.crt and cb-proxy.key and execute :

```
oc delete secret cb-proxy-ssl
oc create secret generic cb-proxy-ssl --from-file=cb-proxy.key --from-file=cb-proxy.crt
```

and restart the cb-proxy POD :

```
oc delete po cb-proxy-xxxx`
```

(with xxxx the name of the POD)

16. Installation of your license Consult the section "Activating-your-License"

17. In case OpenShift cluster is behind a reverse proxy Consult the Configuration section "Collaboard-behind-a-reverse-proxy"

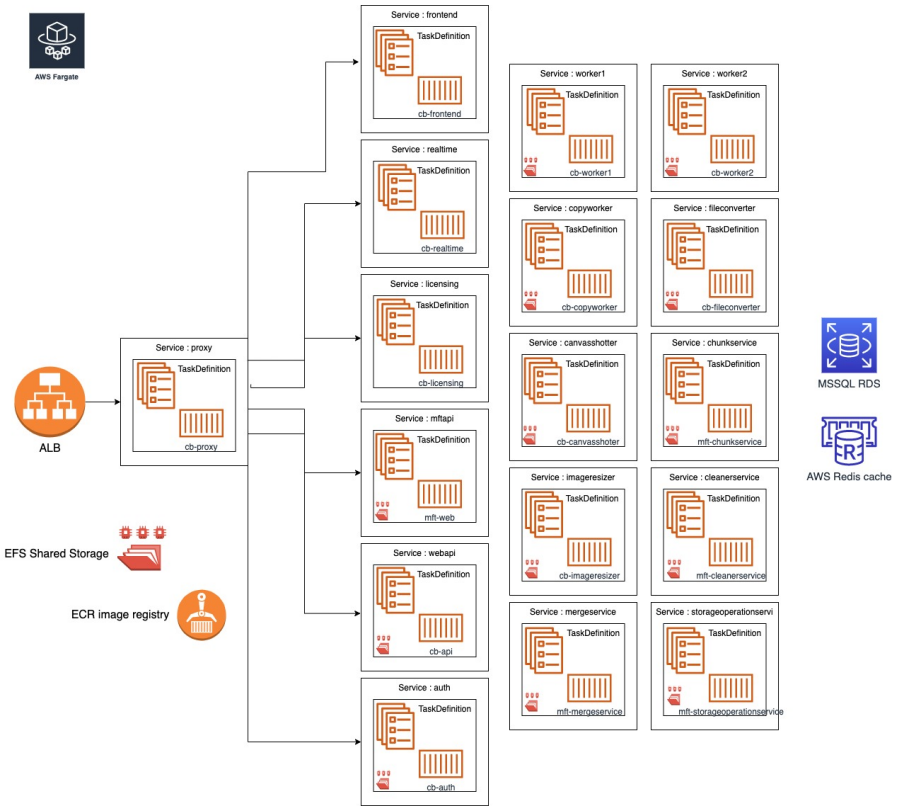
[[*TOC*]]

Scope

This guide describes the installation of Collaboard on AWS ECS Fargate. AWS Fargate is a technology that you can use with Amazon ECS to run containers without having to manage servers or clusters of Amazon EC2 instances. With AWS Fargate, you no longer have to provision, configure, or scale clusters of virtual machines to run containers. This removes the need to choose server types, decide when to scale your clusters, or optimize cluster packing. As this is very specific environment, additional information is provided.

Architecture overview

Each Collaboard container is running in a corresponding ECS Fargate container. The ECS cluster will be deployed in one VPC with only private subnets. Only the cb-proxy container will have access to the "outside" through an ALB loadbalancer. The frontend containers (frontend, auth, api, ...) need to communicate with through a separated ALB loadbalancer which is completely private and internal.



Deployment of the Cluster

The deployment of the ECS Cluster including the Storage, Database and REDIS is done through CloudFormation stacks.

Name	Purpose	Scriptname
IAM	Contains all configured IAM roles	CB-Infra-AWS-IAM.yaml
EFS	Configures all EFS Access Points	CB-Infra-AWS-data-EFS.yaml
RDS	Configures MSSQL RDS database	CB-Infra-AWS-data-RDS.yaml
REDIS	Configures REDIS cache	CB-Infra-AWS-data-REDIS.yaml
ECS	Configures ECS cluster and containers	CB-Infra-AWS-ECS-cluster.yaml

Application Configuration

In order to access the configuration files, you need to deploy a EC2 minimal instance and mount the EFS file-sytem that was created during the deployment.

Filename	Variable	Location
appsettings.Override.json	CB_OVERRIDE_FILE	efs/appsettings
env.Override.js	CB_OVERRIDE_ENV_JS	efs/appsettings
backoffice.Override.js	CB_OVERRIDE_BACKOFFICE	efs/appsettings
ConfigurationMigration.sql	SQL_SCRIPT_FILE	efs/appsettings
envoy.Override.yaml	CB_OVERRIDE_ENVOY	efs/appsettings

Note: Use the template configuration files as a starting point and replace all the placeholders by values that are explained in the Configuration Settings.

Access website on https://

- You might get error code 400 if you try using HTTP. In that case, just refresh the page and it will redirect to HTTPS.

Install SSL certificate

The certificate we use needs to be installed using ACM. We need to ARN when we have it at deployment time of the cluster.

Installation of your license

Consult the section "Activating-your-License"

[[*TOC*]]

Scope

This page describes how to activate/install your license file (license.lic) into the Collaboard application.

Where can I find my license file ?

When you install Collaboard for the first time, you will have a temporary license which has limited features. After your agreement with IBV, you will receive a license file. If not, you can always request it. This license file (license.lic) is custom for your installation.

License file activation/installation

The Collaboard license file needs to be configured in the SQL_SCRIPT_FILE and mounted in the following containers :

- cb_api
- cb_realtime
- cb_auth
- cb_licensing
- cb_mftapi
- cb_copyworker
- cb_worker1
- cb_worker2
- cb_canvasshotter
- cb_fileconverter
- cb_imageresizer
- mft_chunkservice
- mft_cleanerservice
- mft_mergeservice
- mft_storageoperationsservice

Easiest way is to have this file in your folder with overrides (appsettings.Override.json etc.).

Additional SQL scripts (to run ONCE)

There are two additional scripts required to correctly initialize Licensing that need to be run in correct order. Files can be appended to the content of ConfigurationMigration.sql file as they contain the exists check, making sure they are applied **ONLY ONCE**.

Both scripts are part of the Distribution repository in respective "*/sql" folder.

1 - Initialize licensing.sql

File initialized products and other tables.

2 - SelfHostedUser.sql

File creates user with SysAdmin and LicensingManager roles. Make sure you specify correct USERNAME, EMAIL and optionally external authentication provider before applying this script. If you don't specify an external authentication provider, application user with default password will be created.

Default password: `3AFqYFr@CB`

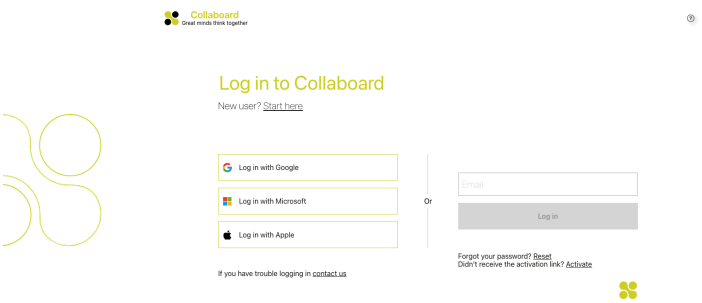
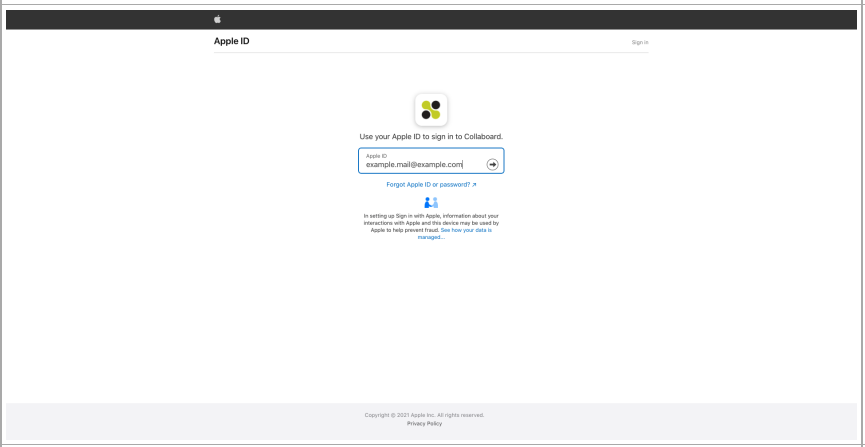
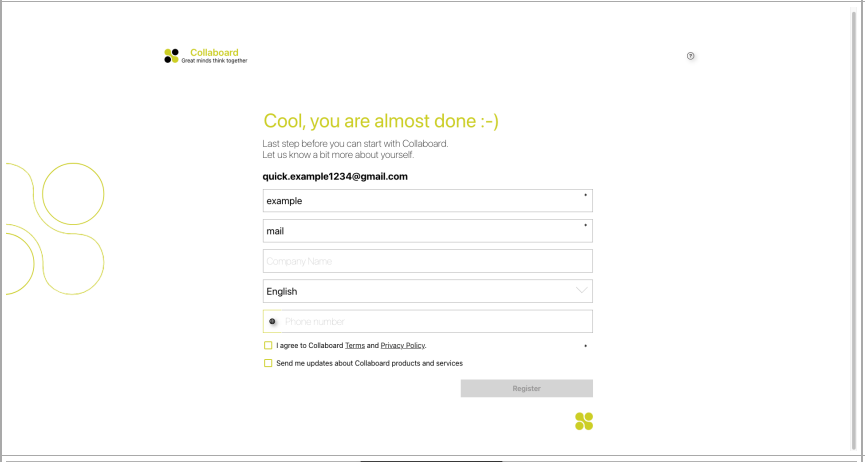
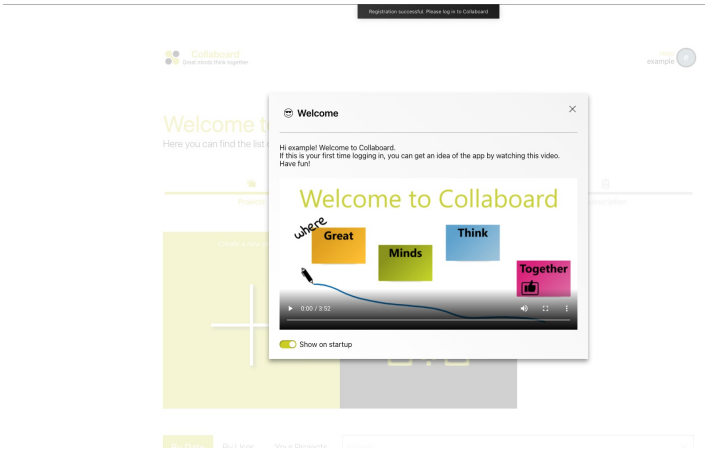
Once the license has been correctly initialized, you can remove these 2 scripts from the SQL_SCRIPT_FILE

[[*TOC*]]

Scope

After a successful installation of all containers you go through the following test procedure in order to check if your application is running as expected.

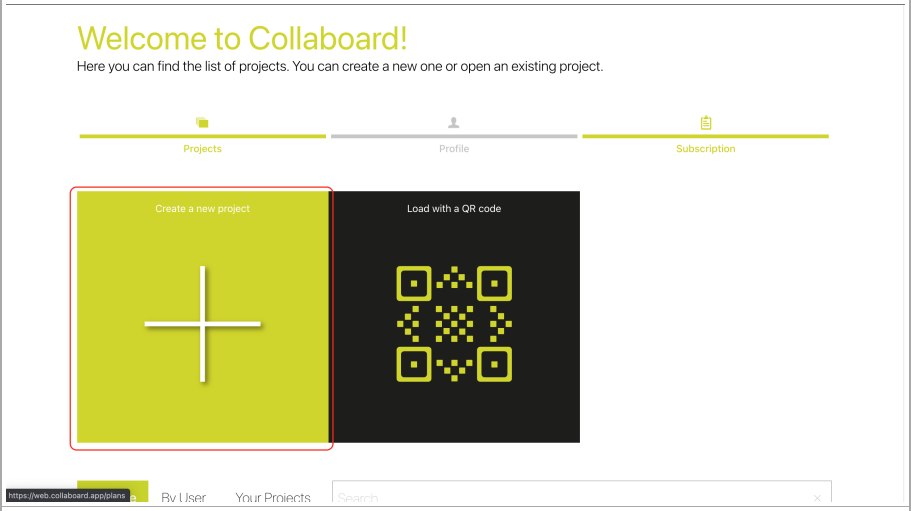
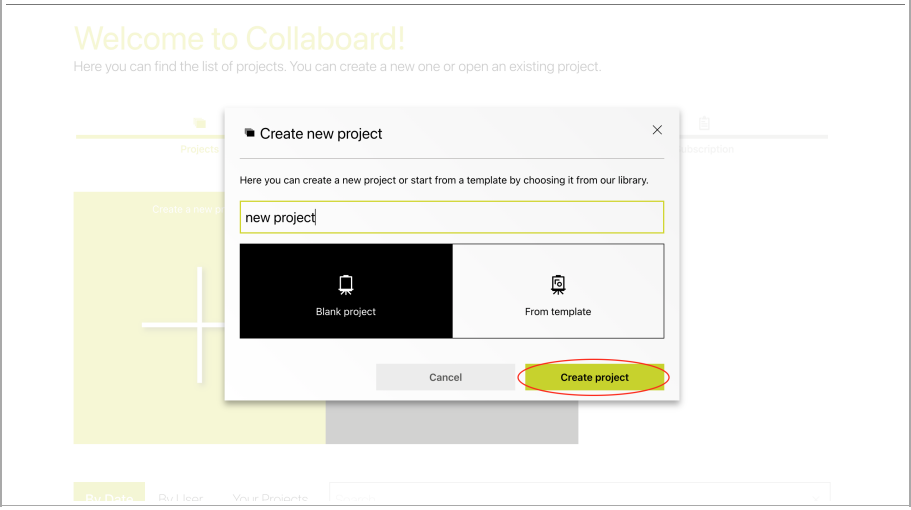
1. Register & Login

Actions required	Screenshots	Expected Result
Navigate to the following URL https://		You should see the Collaboard homepage with all your configured external providers
Click the login button for your desired provider		You should be redirected to the provider sign in page
Complete the sign in on your external provider page		You should be redirected back to Collaboard register page to accept the Terms and Conditions
Check the T&C checkbox and click the "Register" button		You should be logged into your account

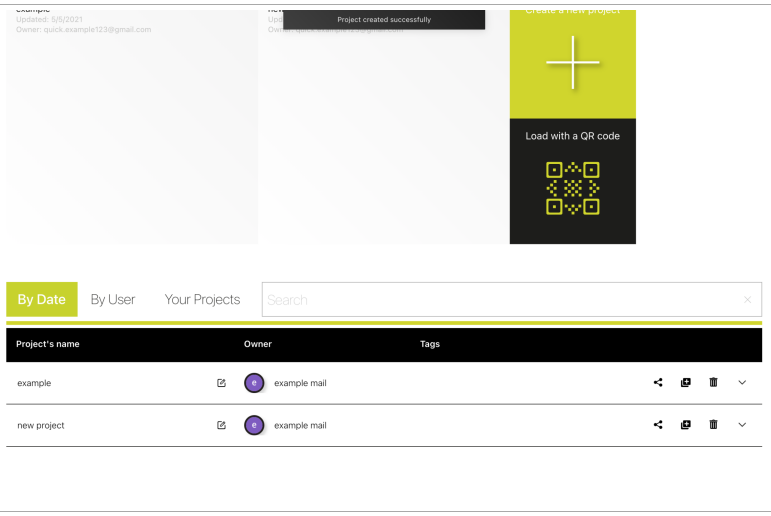
2. Projects page

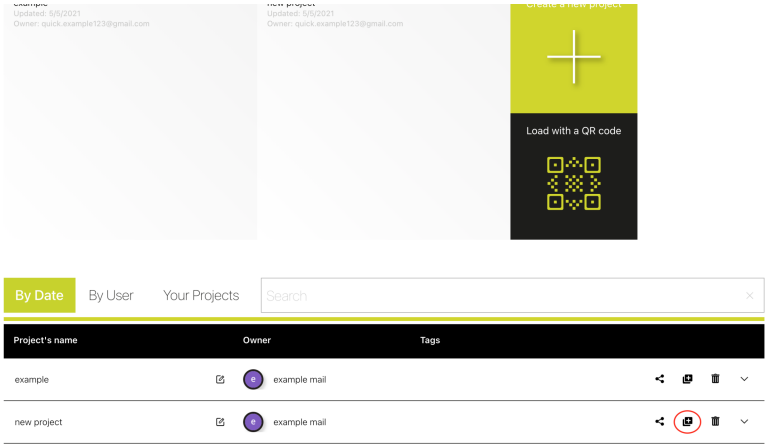
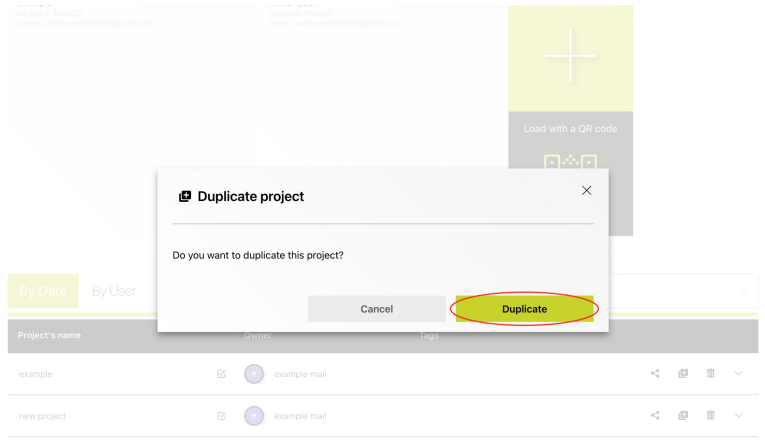
This is the first page you'll see after you Login into your Collaboard account.

2.1 Create a new project

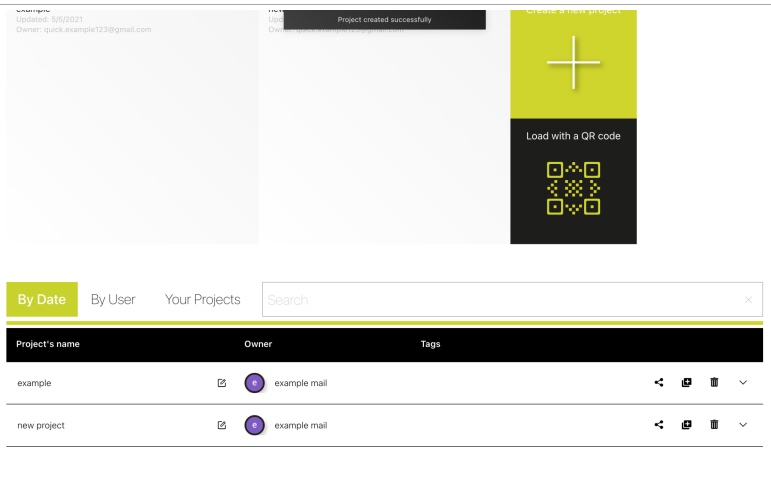
Actions required	Screenshots	Expected Result
Click the "+" button to create a new project		The project creation pop-up should be displayed
Enter a name for your new project and click the "Create project" button		A new project will be created and opened

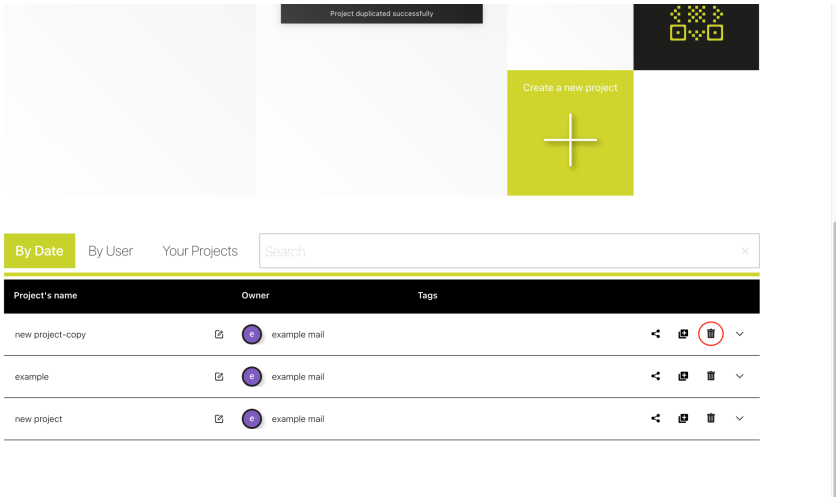
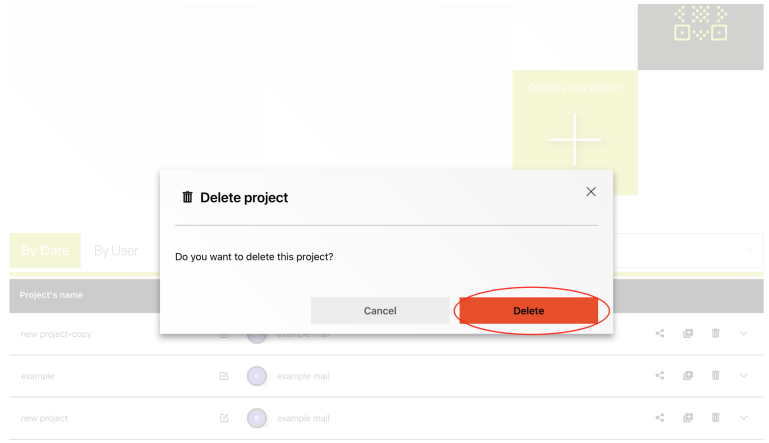
2.2 Duplicate a project

Actions required	Screenshots	Expected Result
Scroll down to see your projects list		Your projects should be displayed each on one row

Actions required	Screenshots	Expected Result
Click the duplicate button located in at the end of the row		A confirmation pop up should appear
Click the "Duplicate" button to confirm the project duplication		The project should be successfully duplicated and opened

2.3 Delete a project

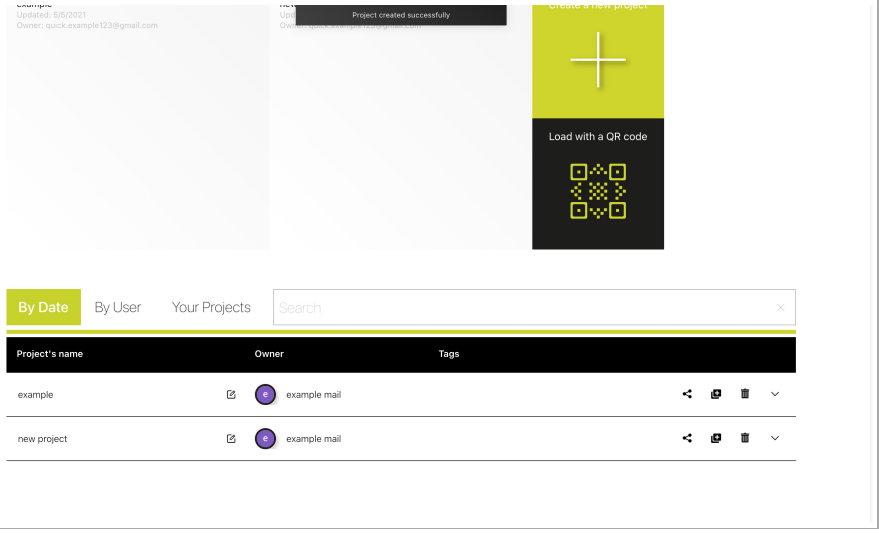
Actions required	Screenshots	Expected Result
Scroll down to see your projects list		Your projects should be displayed each on one row

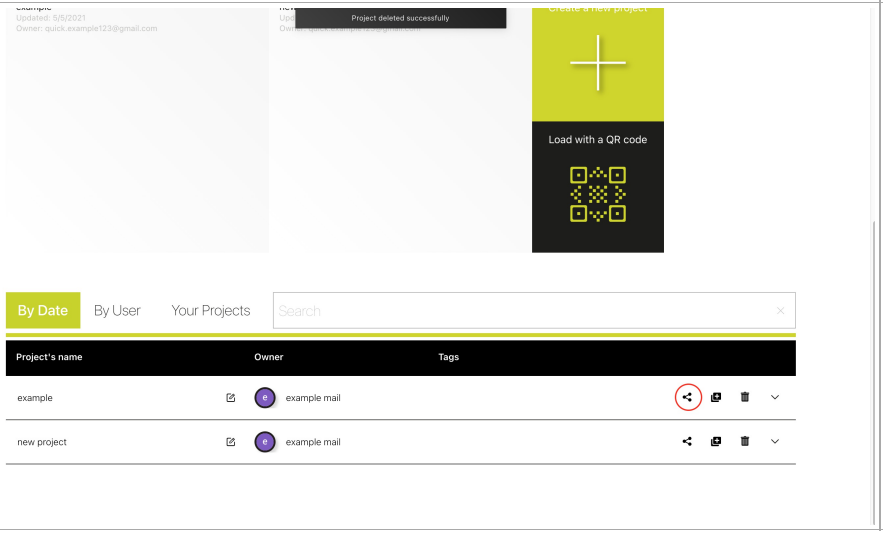
Actions required	Screenshots	Expected Result
Click the delete button located in at the end of the row		A confirmation pop up should appear
Click the "Delete" button to confirm the action		The project should be successfully deleted from your projects list

2.4 Share a project

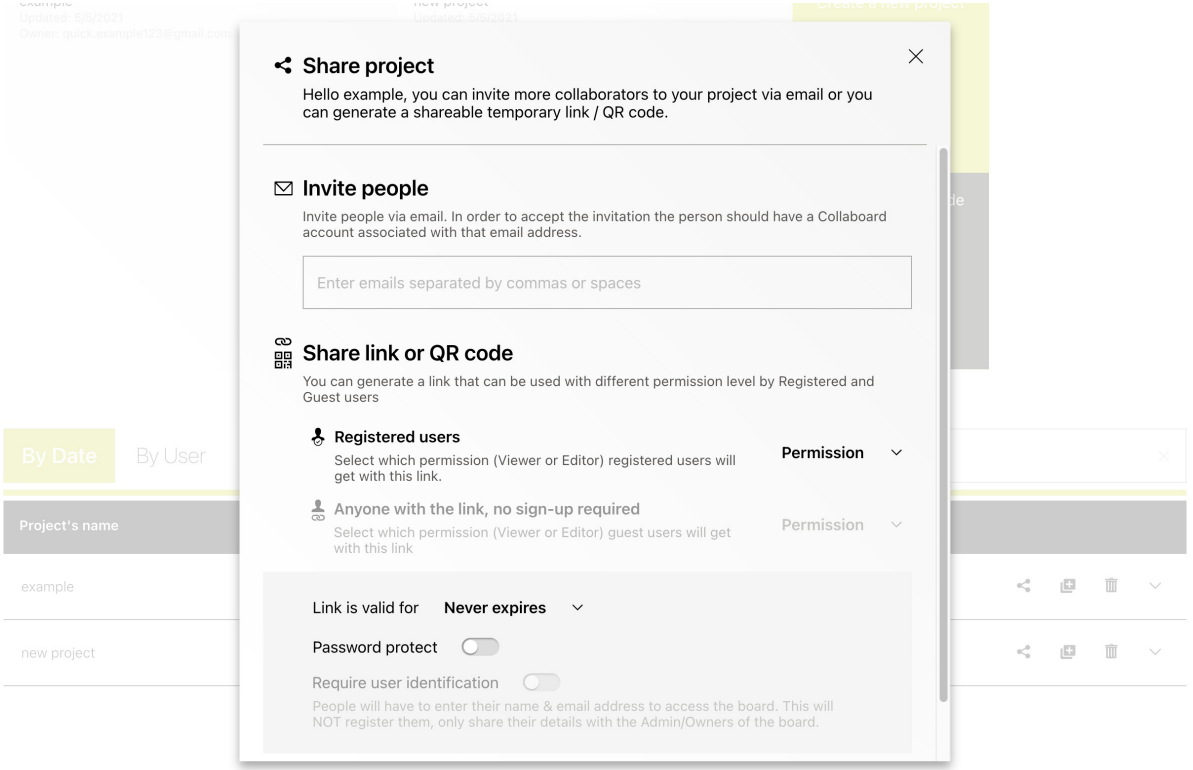
The share project functionality is available here in the projects page, as well as inside a project.

First, lets take a look at the share project pop up:

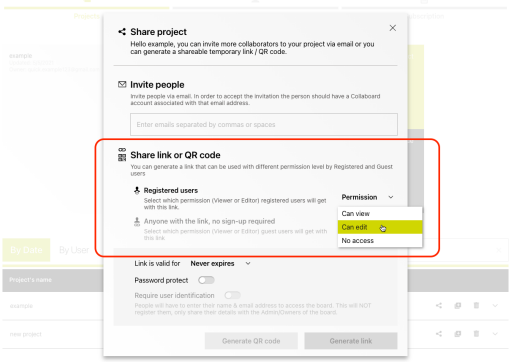
Actions required	Screenshots	Expected Result
Scroll down to see your projects list		Your projects should be displayed each on one row

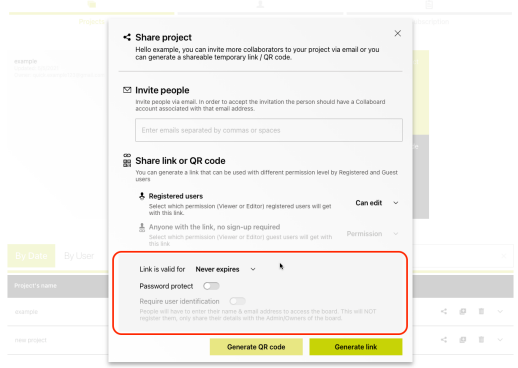
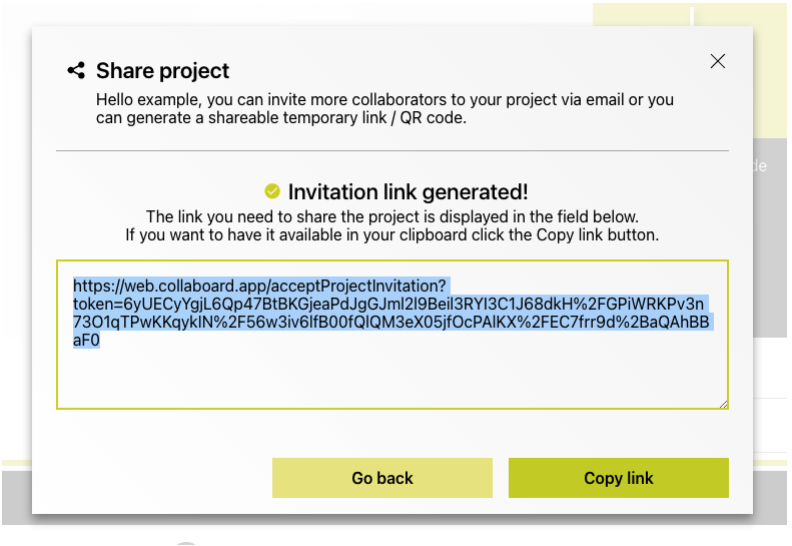
Actions required	Screenshots	Expected Result
Click the share button located in at the end of the row		The share project pop up should appear

The share pop up should look like this:



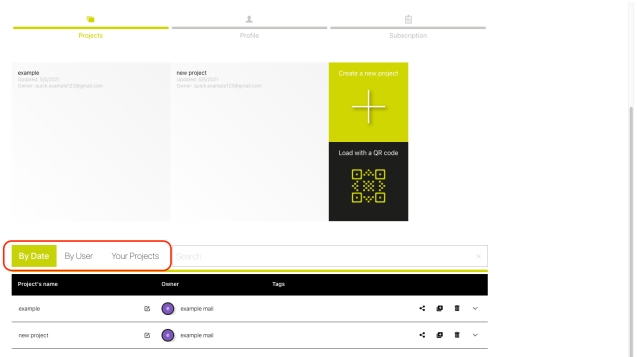
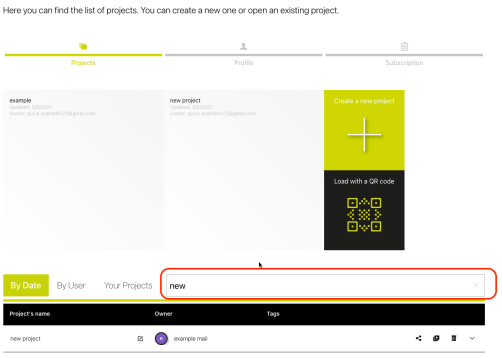
2.4.1 Generated link

Actions required	Screenshots	Expected Result
Select the permissions for the participants		The dropdown should update with the selected permission

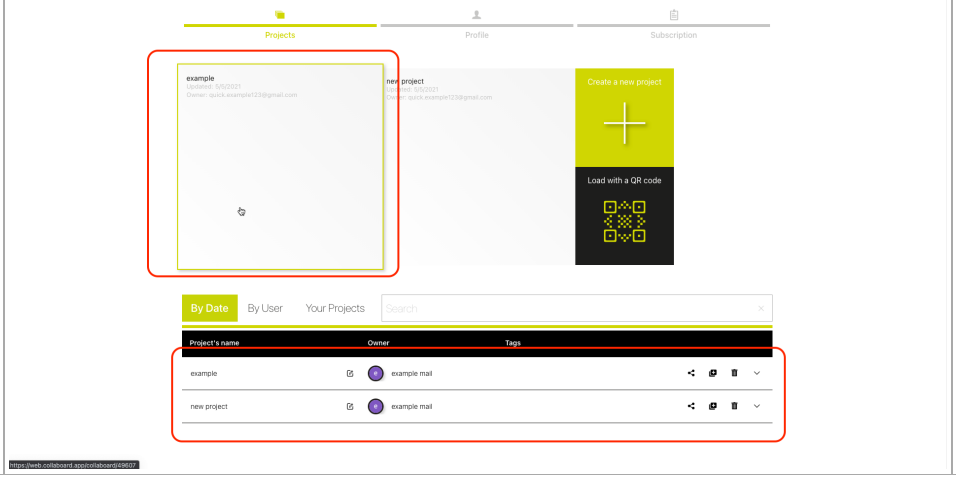
Actions required	Screenshots	Expected Result
Add multiple security options like link lifespan or password protect and check if these apply		-
Click the "Generate link" button		You can now share your project by sending this link to your desired participants

2.5 Filter & Search projects

Actions required	Screenshots	Expected Result

Actions required	Screenshots	Expected Result
You can filter your projects by their date added, by collaborators(users), or you can filter your own projects. Test the tabs located above your projects list		Test if the projects are displayed as expected
Search for a project using the Search bar		Confirm the functionality works

3. Project canvas

Actions required	Screenshots	Expected Result
Click on one of the projects thumbnail, or click a project on the projects list below		The project should load and open

3.1 Share project

The share project process is the same from inside a project, the button is located here:

Make sure to have at least one other participant to your project for the following steps! You can open a new Collaboard session in a different tab or even browser (using another account).

3.2 Toolbar menu

The toolbar menu is located on the left side of your canvas.

Actions required	Screenshots	Expected Result
Use it to select objects, add multiple types of objects like sticky notes, text, documents, images, videos and much more.		Confirm that the functionality works

3.2.1 File upload

In Collaboard, you can add file objects. Here is a short list:

1. Document objects (multiple extensions are supported):

- word
- excel
- powerpoint
- pdf

2. Images:

- from your local computer
- from your devices camera
- from the web

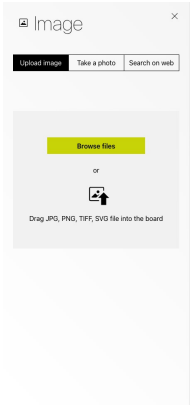
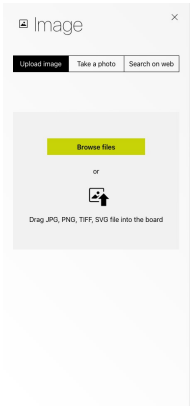
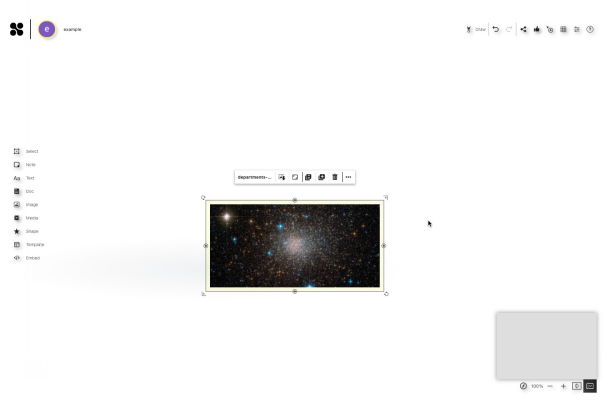
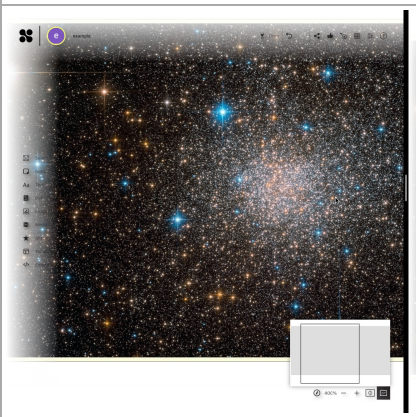
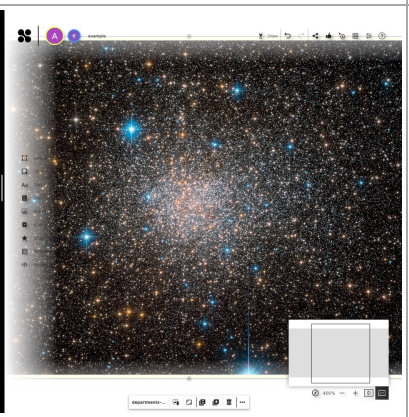
3. Videos:

- from your local computer
- from your devices camera
- youtube videos

3.2.2 Responsive images

Images added on canvas are rendered as responsive ones. This means that you will have multiple images, each at different resolutions, and will change based on your zoom level. Lets add an image to our project.

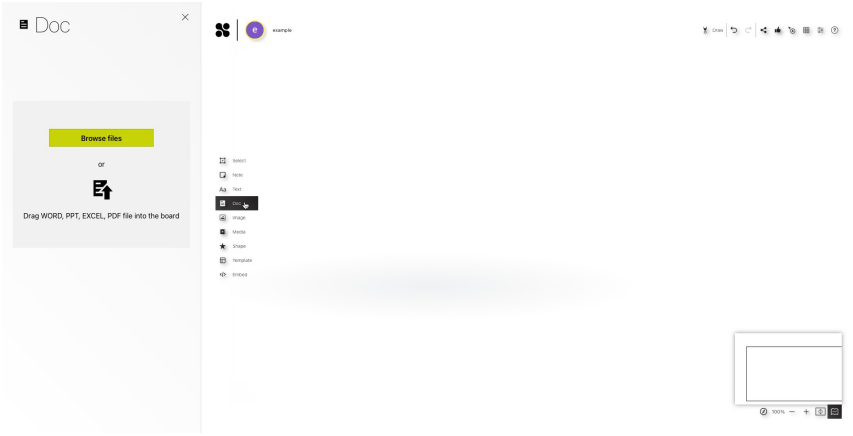
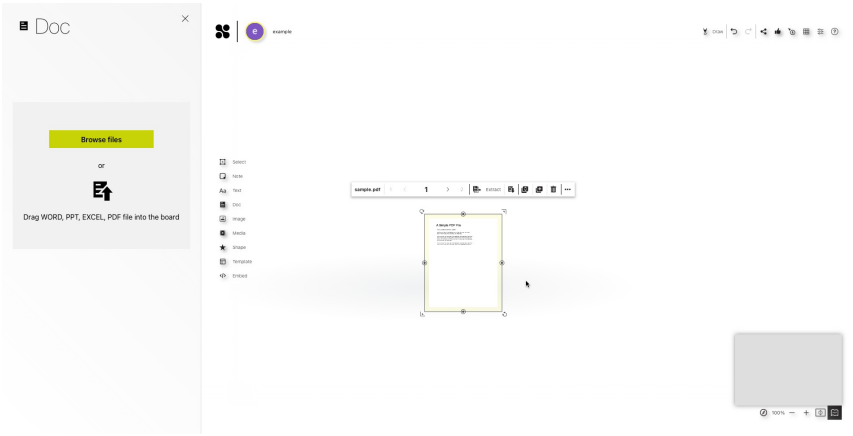
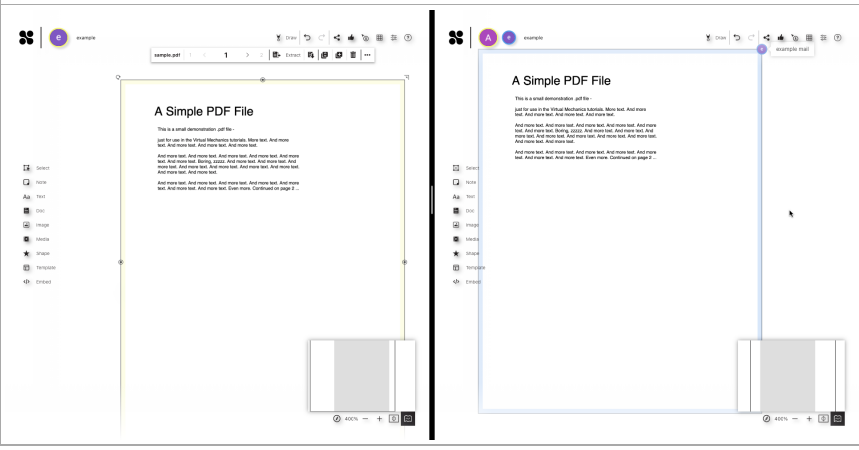
Actions required	Screenshots	Expected Result

Actions required	Screenshots	Expected Result
Click the "Image" button from the toolbar menu	 	The image side bar should open
Upload an image from your local computer	 	The images should be added to the canvas and the server should generate the responsive images
Make sure the image is rendered for every participant, zoom in and out to check	 	The images should be properly rendered depending on your zoom level

3.2.3 Document Objects

Uploaded documents should have a thumbnail generated where you can preview its pages live on canvas

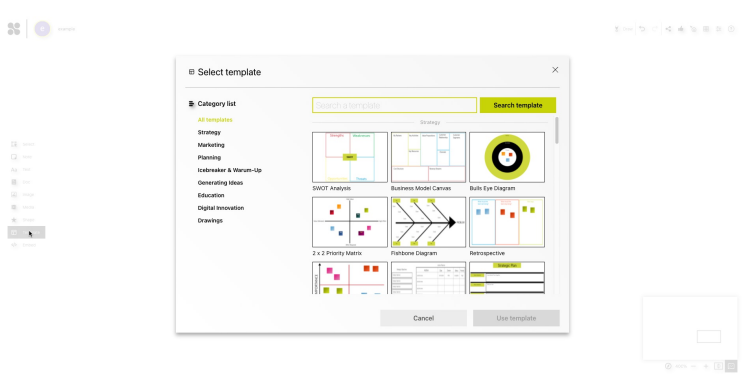
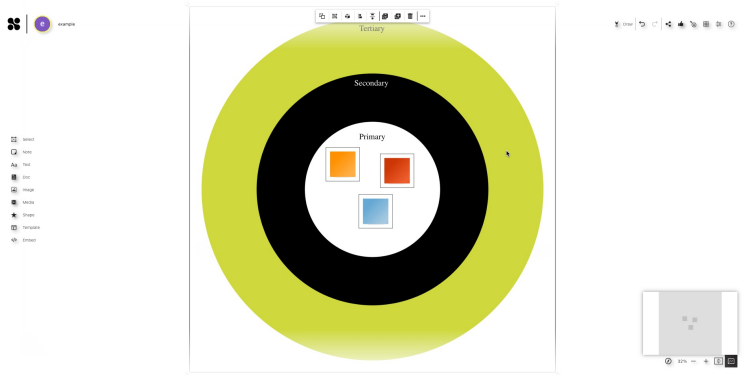
Actions required	Screenshots	Expected Result

Actions required	Screenshots	Expected Result
Click the "Document" button from the toolbar menu		The document side bar should open
Upload a document from your local computer		Previews should be generated for every page
Make sure the document thumbnail is rendered for every participant		The document thumbnails should be responsive too, depending on your zoom level

3.2.5 Templates

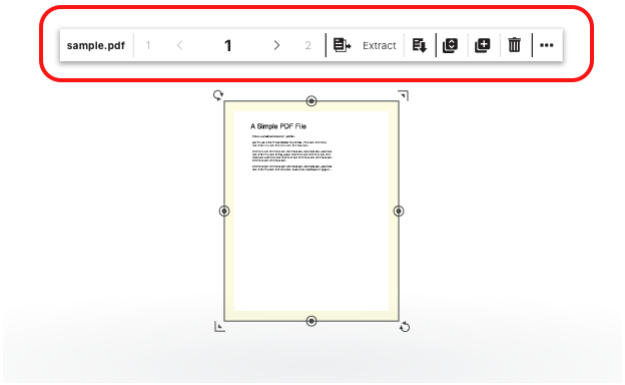
Let's add a template to the canvas. Here you can find multiple categories you can choose from.

Actions required	Screenshots	Expected Result

Actions required	Screenshots	Expected Result
Click the "Template" button from the toolbar menu		The template pop up should appear
Choose a template and add it on canvas by using the "Add" button or by double clicking on its thumbnail		The template should be added on canvas and you will automatically be zoomed in the center of it

3.3 General object features (context menu)

At this point you most likely encountered the context menu. To open it, select an object, it should look like this:



From here you can make multiple actions, depending on the type of object you have selected.

Actions required	Screenshots	Expected Result
Duplicate an object by clicking the "Duplicate" button		The object should be copied and rendered on canvas
Delete an object by clicking the "Delete" button		Selected object should be removed from the canvas
Navigate through pages of a document		As you navigate, the pages should be displayed in the thumbnail
Extract single or multiple pages of a document		Pages extracted will be added on canvas as images (make sure to check if they are responsive)
Download any type of file (image, document, video) using the "Download" button		The file should be successfully downloaded to your local computer

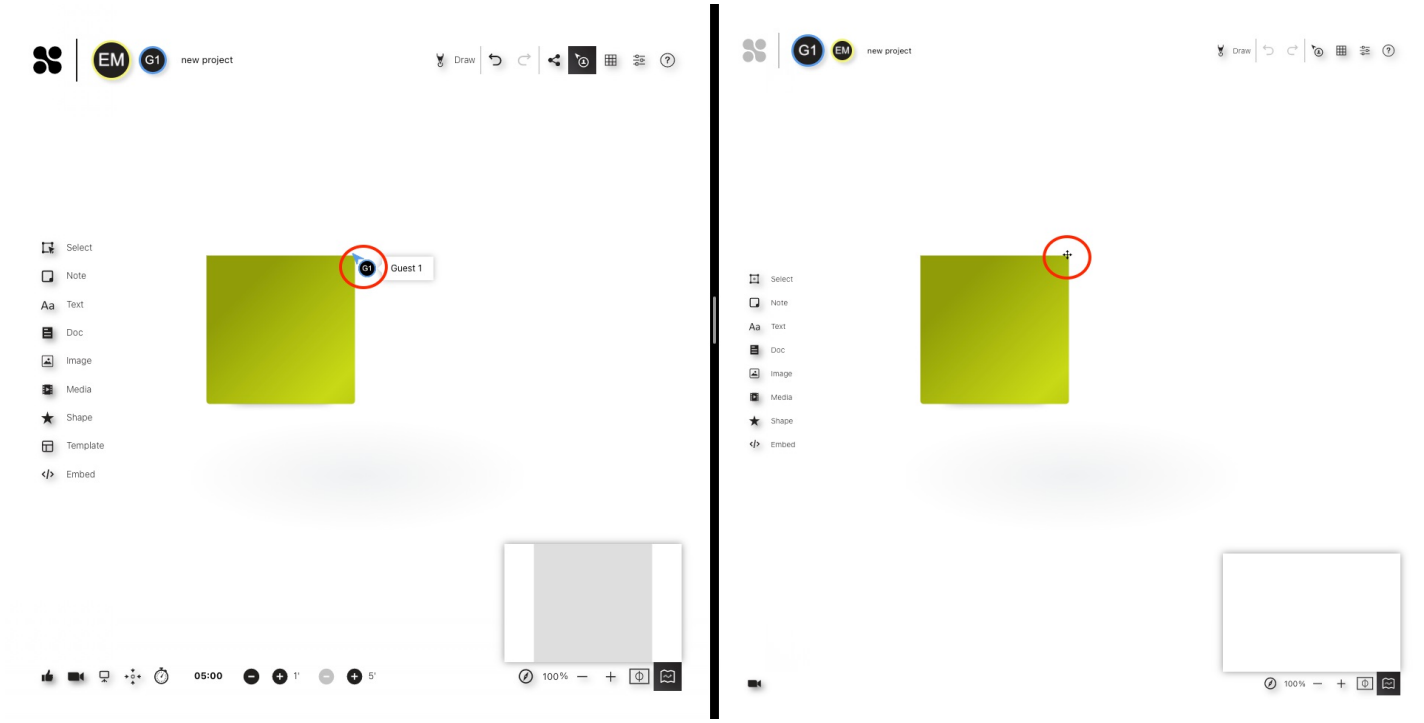
Collaboard supports keyboard shortcuts. For example you can duplicate an object by selecting it and hitting ctrl/cmd + c on your keyboard, delete it by hitting the backspace, select all objects using ctrl/cmd + a etc.

3.4 User Presence

Actions required	Screenshots	Expected Result

Actions required	Screenshots	Expected Result
Activate User presence using the button from the upper right corner		Make sure the cursors of other participants are displayed on your canvas and are accurate

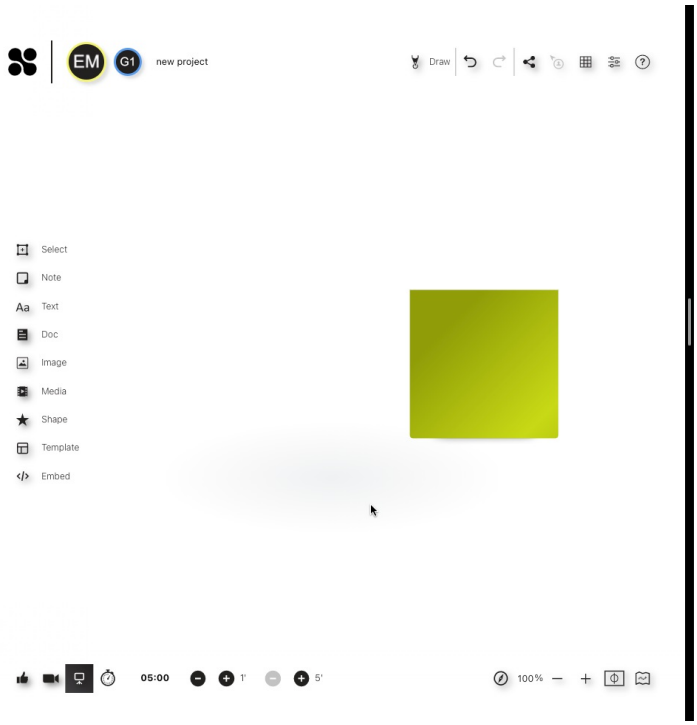
It should look like this:

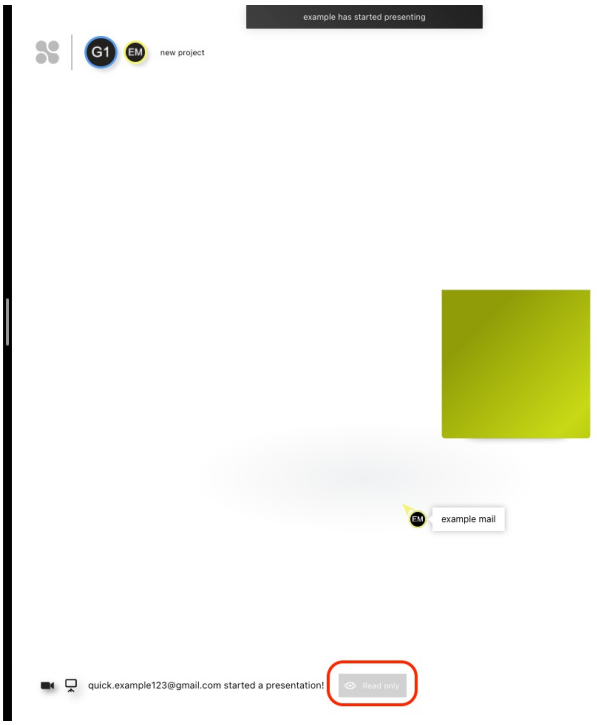


3.5 Presentation mode

Actions required	Screenshots	Expected Result
Activate Presentation mode using the button from the bottom left corner		Participants view should be the same as the presentator view

While presentation mode is active, participants should be "Read only". Make sure changes made by the presentator are reflected on the rest of the participants canvas.


EM
G1
new project
Draw
Select
Note
Text
Doc
Image
Media
Shape
Template
Embed
05:00
100%


example has started presenting
G1
EM
new project
example mail
quick.example123@gmail.com started a presentation!
Read only

[[*TOC*]]

Overview

In total, there are 2 ways to scale the environment.

- Services scaling *Each background service has scalable threads count (parallelization inside container) and delay between subsequent operations.*
- Containers scaling *Most of the containers can be scaled both horizontally and vertically. Containers cb_proxy, cb_coturn, cb_db and redis currently support only vertical scaling.*

[[*TOC*]]

Overview

Each background service is running as bundle of threads which are created on application startup. These threads are checking respective operation queue with configured delay. Each service has it's set of values in **dbo.t_configuration** table.

- Thread count -- defines how many threads are created on application startup = how many operations can be running asynchronously on respective container.
- Thread delay -- defines the delay between consecutive requests to database queue.
- Retry count -- max number of retries before operation is dropped as Failed.

Default service scaling script with consideration to the container scaling policy is in progress and will be provided. However, due to differences in hardware or hosting environment power, auto-scaling policies need to be tested.

List

Following is the list of all the keys used to scale services.

Async services

- Server_CanvasShotterDelayInThreads
- Server_CanvasShotterMaxRetries
- Server_CanvasShotterMaxThread

- Server_CopiedTilesDelayInThread
- Server_CopiedTilesMaxThread

- Server_CopyProjectDelayInThread
- Server_CopyProjectMaxRetries
- Server_CopyProjectMaxThread

- Server_CopyProjectFBDelayInThread
- Server_CopyProjectFBMaxThread

- Server_CopyProjectHistMaxThread
- Server_CopyProjectHistThreadDelay

- Server_CopyTileDelayInThread
- Server_CopyTileMaxRetries
- Server_CopyTileMaxThread

- Server_CopyTileFBDelayInThread
- Server_CopyTileFBMaxThread

- Server_CopyTileHistoryMaxThread
- Server_CopyTileHistoryThreadDelay

- Server_LicImportDelayInThread
- Server_LicImportMaxThread

- Server_LicPaymentsDelayInThread
- Server_LicPaymentsMaxThread

- Server_ShotterFallbackDelayInThread
- Server_ShotterFallbackElapsedTime
- Server_ShotterFallbackMaxThread

- Server_ShotterHistoryDelayInThread
- Server_ShotterHistoryMaxThread

- Server_TileInspDelayInThread
- Server_TileInspMaxThread

- Server_UserInspectorDelayInThread
- Server_UserInspectorMaxRetries
- Server_UserInspectorMaxThread

- Server_UserInspFbDelayInThread
- Server_UserInspFbMaxThread

- Server_FileConvDelayInThread
- Server_FileConvMaxRetries
- Server_FileConvMaxThread

- Server_FileConvFallbackMaxThread
- Server_FileConvFallbackThreadDelay

- Server_FileConvHistoryMaxThread
- Server_FileConvHistoryThreadDelay

- Server_ImageHistoryMaxThread
- Server_ImageHistoryThreadDelay

- Server_ImgFallbackDelayInThread
- Server_ImgFallbackMaxThread

- Server_ImgResizeDelayInThread
- Server_ImgResizeMaxRetries
- Server_ImgResizeMaxThread

- Server_TempCleanerMaxThread
- Server_TempCleanerThreadDelay

MFT Services

- ChunkingMaxRetries
- ChunkingServiceConcurrentThreads
- ChunkingServiceSleepTimeInMillisecs

- MergingMaxRetries
- MergingServiceConcurrentThreads
- MergingServiceSleepTimeInMillisecs

- StorageMaxRetries
- StorageServiceConcurrentThreads
- StorageServiceSleepTimeInMillisecs

- CleaningMaxRetries
- CleaningServiceConcurrentThreads

- CleaningServiceSleepTimeInMillisecs

Scaling script

We have provided you a scaling script in the repository to help you scale the Collaboard application in order to make optimal use of your environment and have better performances. The scaling script (called ***4_ScaleScript.sql***) is a SQL script that will update the scaling parameters in the SQL database.

Installation steps

- in case you have access to the database via a SQL editor (e.g. SQL Server Management Studio), you can simple run the script into the editor.
- in case you do not have direct access to the SQL database, you can run the script using the DB-INIT method. This can be done through a configmap.

a. Check the database name in the 4_ScaleScript.sql script and update if needed.
b. Create a configmap called "cb-scaling-script" `kubectl create cm cb-scaling-script --from-file=ConfigurationMigration.sql=4_ScaleScript.sql`
c. Create a new job to mount the configmap with the SQL script and run it against the DB `kubectl create -f cb-job-scaling.yaml`
d. Check the logs output of the POD for a successful completion.

[[*TOC*]]

Overview

The container scaling is based on the orchestrator capabilities.

Docker Compose

Docker compose has limited scaling capabilities using the command :

```
docker-compose scale <service name> = <no of instances>

example : docker-compose scale cb-api=3
```

Kubernetes and OpenShift

Besides the manual scaling capabilities, it has also a horizontal and vertical auto POD scaler (HPA vs VPA). More info on this can be found here : https://kubernetes.io/blog/2016/07/autoscaling-in-kubernetes/

- Manual :

```
kubectl scale deployment <service name> --replicas=<no of instances>

example : kubectl scale deployment cb-api --replicas=3
```

- Autoscaling :

```
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
  annotations:
    autoscaling.alpha.kubernetes.io/conditions: '[{"type":"AbleToScale","status":"True","lastTransitionTime":"2021-02-01T11:40:12Z","reason":"ReadyForNewScale","message":"recomme
      size matches current size"},{"type":"ScalingActive","status":"True","lastTransitionTime":"2021-02-01T17:30:50Z","reason":"ValidMetricFound","message":"the
      HPA was able to successfully calculate a replica count from cpu resource utilization
      (percentage of request)"},{"type":"ScalingLimited","status":"False","lastTransitionTime":"2021-02-01T11:40:12Z","reason":"DesiredWithinRange","message":"the
      desired count is within the acceptable range"}]'
    autoscaling.alpha.kubernetes.io/current-metrics: '[{"type":"Resource","resource":{"name":"cpu","currentAverageUtilization":3,"currentAverageValue":"3m"}}]'
  name: cb-api
spec:
  maxReplicas: 3
  minReplicas: 1
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: cb-api
  targetCPUUtilizationPercentage: 60
```

Further examples can be found in the horizontalpodautoscalers subdirectory in the Distribution repository.

Collaboard container scaling

All background container services can be scaled depending on the CPU usage (e.g. 70%). The app comes already configured with the following scaling policies : Container|Minimum|Maximum|CPU Requests|CPU Limits|Memory Requests|Memory Limits --|--|--|--|--|--|-- cb-copyworker|1|5|150m|500m|64Mi|700Mi cb-fileconverter|1|8|150m|800m|64Mi|2200Mi cb-imageresizer|1|5|150m|800m|64Mi|2200Mi cb-realtime|1|2|50m|500m|64Mi|1000Mi cb-worker1|1|2|50m|600m|64Mi|500Mi cb-worker2|1|2|50m|600m|64Mi|500Mi mft-chunkservice|1|2|--|--|--|-- mft-storageoperationsservice|1|2|--|--|--|-- mft-cleanerservice|1|2|--|--|--|-- cb-canvasshotter|1|3|50m|600m|64Mi|2000Mi mft-mergeservice|1|2|--|--|--|-- cb-api|1|2|50m|400m|64Mi|600Mi cb-auth|1|2|--|--|--|-- mft-web|1|2|--|--|--|-- cb-licensing|1|2|--|--|--|-- cb-proxy|1|2|--|--|--|-- cb-frontend|1|2|--|--|--|--

Important Note : Do not configure the DB with autoscaling as it will cause a corruption of your data

[[*TOC*]]

@<763D03B3-764A-6271-8DF6-64569226E05C> @<59B13A69-0456-6853-B560-84A0D40ECAFB> In this chapter, we shall describe common issues our customers reported, and how to solve.

1. Login issue

Using multiple DevOps

Symptoms:

- User is getting **401 - Uh-oh, you do not have access** even though the access has been set up.

Possible solution:

- If user is using DevOps of another organization, then there is probably a clash of cookies. Solutions:
  - Use a different browser that is not used to access any other DevOps organization.
  - Use anonymous mode.
  - Log out or clean cookies in the existing one.

2. Collaboard PODs are in CrashLoopBackOff status

This happens only on Kubernetes or OpenShift

Symptoms: When the POD gets started it immediately crashes and goes into the CrashLoopBackOff state

Troubleshooting: There can be several reasons for this, but most likely it is because the PODs can not connect to the database. You can connect to one of the PODs that is running and test the DB connectivity.

```
kubectl exec -it <POD-name> -- sh
curl db:1433
```

This should return immediately the prompt, if not it means there is no connectivity to the DB.

Possible solution:

- make sure the DB service has been configured. If not, create it with the command

```
kubectl create -f services/
```

Here is the list of breaking changes and steps to undertake when migrating existing system. Changes shall listed in order from newest to oldest.

[[*TOC*]]

1. New features

- Storage client library integration
- Backoffice
- Project templates
- Enhanced security

2. Important Note before you start

It is extremely important that you take a backup of Collaboard before you start.

- Backup of ALL persistent volumes. Make sure the application is stopped before doing that, otherwise it will not be consistent)
- Backup of the MS SQL database in case it runs externally. You might need this in case you fail the migration as this upgrade will not be compatible with any previous versions.

3. Update procedure

Files to update:

env.Override.js

```
{
  ...
  storageProvider: "MFT",
  storageUploadChunkSize: 128000,
  storageDownloadChunkSize: 128000,
  ...
  features: {
    ...
    storageClient: true,
    disableProjectKey: true,
    createProjectToS: true,
    noEcommerce: true,
    ...
  }
  ...
}
```

ConfigurationMigration.sql

```
declare @GenericAppDomain nvarchar(50) = '*'

...
,('Server_MFTClientBufferSize', '< PROVIDE VALUE >', 'Chunksize to use with MFT', @GenericAppDomain)
,('Server_MFTSignatureCryptexKeys', '< PROVIDE VALUE >', 'MFT signature enctyption keys', @GenericAppDomain)
...
```

If Azure Blob Storage is being used, then AppDomain these three keys need to be set to '*':

- Server_AccountKey
- Server_AccountName
- Server_BaseUriWrite

envoy.Override.yaml

There are two steps for envoy.Override.yaml.

First step is to add backoffice cluster declaration among clusters (near the end of the file):

```
  - name: backoffice_cluster
    connect_timeout: 30s
    type: LOGICAL_DNS
    # Comment out the following line to test on v6 networks
    dns_lookup_family: V4_ONLY
    lb_policy: round_robin
    load_assignment:
      cluster_name: no1
      endpoints:
      - lb_endpoints:
        - endpoint:
            address:
              socket_address:
                address: backoffice
                port_value: 8080
```

Second step is to add the routing, where you can choose between "port" or "url" based traffic routing.

A. Port based routing

Add following listener. It will be listening on port 9443 and routing all traffic to the Backoffice cluster.

Listener:

```
  - name: backoffice_listener
    address:
      socket_address:
        address: 0.0.0.0
        port_value: 9443
    listener_filters:
    - name: "envoy.filters.listener.tls_inspector"
      typed_config: {}
    filter_chains:
    - filters:
      - name: envoy.filters.network.http_connection_manager
        typed_config:
          "@type": type.googleapis.com/envoy.extensions.filters.network.http_connection_manager.v3.HttpConnectionManager
          stat_prefix: ingress_http
          # access_log:
          # - name: envoy.access_loggers.file
          #   typed_config:
          #     "@type": type.googleapis.com/envoy.extensions.access_loggers.file.v3.FileAccessLog
          #     path: /dev/stdout
          http_filters:
          - name: envoy.filters.http.router
          route_config:
            name: local_route
            virtual_hosts:
            - name: main_routing
              domains: ["*"]
              routes:
                - match:
                    prefix: "/"
                  route:
                    cluster: backoffice_cluster
      transport_socket:
        name: envoy.transport_sockets.tls
        typed_config:
          "@type": type.googleapis.com/envoy.extensions.transport_sockets.tls.v3.DownstreamTlsContext
          common_tls_context:
            tls_certificates:
              certificate_chain:
                filename: /etc/envoy/certs/cb-proxy.crt
              private_key:
                filename: /etc/envoy/certs/cb-proxy.key
            alpn_protocols: [ "h2,http/1.1" ]
```

B. Url based routing

For this kind of routing you need to update the existing listener by adding new routing under **virtual_hosts**. In the following example, any request made to URL beginning with *"/admin."* will be routed through the **backoffice_routing** to

backoffice_cluster. The rest is handled by the **main_routing**.

```
static_resources:
  listeners:
  - name: main_listener
    ...
    filter_chains:
    - filters:
      - name: envoy.filters.network.http_connection_manager
        typed_config:
          ...
          route_config:
            ...
            virtual_hosts:
            - name: backoffice_routing
              domains: ["admin.*"]
              routes:
                - match:
                    prefix: "/"
                  route:
                    cluster: backoffice_cluster
            - name: main_routing
              domains: ["www.*"]
              routes:
                - match:
                    prefix: "/server/signalr"
                  route:
                    prefix_rewrite: "/signalr"
                    ...
```

###backoffice.Override.js This new override file can be fetched from Distribution repository in the overrides folder. Add this to your overrides folder.

```
window.runtimeConfig = {
  apiUrl: "https://localhost:8443/server",
  appName: "webapp-admin",
  authProviders: [],
  pages:{
    configurationSettings: true,
    messageTemplates: true,
    onlineUsers: true,
    products: true,
    projects: true,
    projectTiles: true,
    subscriptionUsers: true,
    templateCategories: true,
    templateProjects: true,
    users: true
  }
}
```

If you are using any external authentication providers, prepare new set for respective appName of backoffice.Override.js with proper redirects to the Backoffice. Configuration follows the same pattern as Collaboard app providers.

New Containers to deploy :

Backoffice

New container cb_backoffice has been added. New override file called **backoffice.Override.js** is required among overrides and it needs to be injected the same way as env.Override.js is being injected to the cb_frontend container.

- **Within docker-compose**
  - Add following snippet to the docker-compose.yml

```
  backoffice:
    depends_on:
      - api
      - coturn
    image: ${ENV_REGISTRY}/cb_backoffice:${ENV_VERSION}
    volumes:
      - ${CB_OVERRIDES}:/var/overrides
    environment:
      - CB_OVERRIDE_BACKOFFICE=${CB_OVERRIDE_BACKOFFICE}
    stdin_open: true
    tty: true
    container_name: cb_backoffice
    restart: unless-stopped
    networks:
      - cbnetwork
```

- Add following line to the .env file

```
CB_OVERRIDE_BACKOFFICE=backoffice.Override.js
```

- **Within Kubernetes/OpenShift** The backoffice manifest file can be fetched from Distribution repository in the deployments folder.

```
kubectl create -f cb-backoffice.yaml
```

Project Upgrader

New container cb_projectupgrader has been added. It shares overrides with other worker containers, i.e. cb_copyworker.

- **Within docker-compose**
  - Add following snippet to the docker-compose.yml

```
  projectupgrader:
    depends_on:
      - db
    image: ${ENV_REGISTRY}/cb_projectupgrader:${ENV_VERSION}
    volumes:
      - ${CB_FILES}:/var/mft/_cbfiles_
      - ${CB_TEMP}:/var/mft/_temp_
      - ${CB_OVERRIDES}:/var/overrides
    environment:
      - CB_OVERRIDE_FILE=${CB_OVERRIDE_FILE}
    container_name: cb_projectupgrader
    restart: unless-stopped
    networks:
      - cbnetwork
```

- **Within Kubernetes/OpenShift** The projectupgrader manifest file can be fetched from Distribution repository in the deployments folder.

```
kubectl create -f cb-projectupgrader.yaml
```

Upgrade Fallback

New container cb_upgradefallback has been added. It shares overrides with other worker containers, i.e. cb_copyworker.

- **Within docker-compose**
  - Add following snippet to the docker-compose.yml

```
  upgradefallback:
    depends_on:
      - db
    image: ${ENV_REGISTRY}/cb_upgradefallback:${ENV_VERSION}
    volumes:
      - ${CB_FILES}:/var/mft/_cbfiles_
      - ${CB_TEMP}:/var/mft/_temp_
      - ${CB_OVERRIDES}:/var/overrides
    environment:
      - CB_OVERRIDE_FILE=${CB_OVERRIDE_FILE}
    container_name: cb_upgradefallback
    restart: unless-stopped
    networks:
      - cbnetwork
```

- **Within Kubernetes/OpenShift** The upgrade-fallback manifest file can be fetched from Distribution repository in the deployments folder.

```
kubectl create -f cb-upgradefallback.yaml
```

Update existing Containers with new tag :

- **Within docker-compose** Update the following line in the .env file

```
ENV_VERSION=5.0.4838
```

Run the script **update-and-run.ps1** or **update-and-run.sh** Make sure you press "y" to update the container images

- **Within Kubernetes/OpenShift**

```
kubectl get deploy | while read DEPLOY B; do   UDEPLOY=$(echo $DEPLOY | tr '-' '_') ;  kubectl set image deploy/$DEPLOY *=ibvcollaboard.azurecr.io/$UDEPLOY:5.0.4838; done
```

Update the database to the latest version :

- **Within docker-compose** Update the following line in the .env file

```
ENV_VERSION=5.0.4838
```

Run the script **update-and-run.ps1** or **update-and-run.sh** Make sure you press "y" to update the DB

- **Within Kubernetes/OpenShift**

```
cd jobs
# update image reference in cb-db-init-job.yaml with latest tag 5.0.4838
kubectl delete job db-init
kubectl create -f cb-db-init-job.yaml
```

Make sure the job was successfully completed.

Collaboard Project Templates

!! IMPORTANT note !! These steps can only be used in case there are no existing templates. If there are any existing project templates, you need to remove these first in the Backoffice interface.

The following instructions provide a way to install the standard project templates in a Collaboard deployment. The installation will run a SQL script against the MS SQL database. The script is provided in the repo with the name "3_install_templates.sql".

- in case you have access to the database via a SQL editor (e.g. SQL Server Management Studio), you can simple run the script into the editor.
- in case you do not have direct access to it, you can run the scripts using the DB-INIT method. It can not be done through a configmap as the SQL script is bigger than 2Mb. Therefor a new persistent volume needs to be created. These are the steps :

a. Check the database name in the 3_install_templates.sql script and update if needed.
b. Create a persistent volume called "cb-templates" with size 1GB `kubectl create -f cb-pvc-templates.yaml`
c. Deploy a toolbox POD that mounts this PVC `kubectl create -f toolbox.yaml`
d. Copy the SQL script into the POD `kubectl cp 3_install_templates.sql  toolbox-xxx:/var/tempc` with xxx the corresponding POD-id
e. Delete the toolbox POD `kubectl delete -f toolbox.yaml`
f. Create a new job to mount the volume with the SQL script and run it against the DB `kubectl create -f cb-job-templates.yaml`
g. Check the logs output of the POD for successfull completion and for the list of the directories to create.

- Create the directories in /var/mft/*cbfiles* which is present on most of the PODs (e.g. cb-worker1-xxx) and docker-compose containers (e.g. cb_worker1)
  1. Connect to container:
     - Kubernetes/OpenShift

     ```
     kubectl exec -it cb-worker1-xxx sh
     ```

     - DockerCompose

     ```
     docker exec -it cb_worker1 sh
     ```

  2. Execute script inside of container:

     ```
     cd /var/mft/_cbfiles_
     for DIR in 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49
     do
       DIR_TO_CREATE=$(printf "%03d\n" $DIR)
       mkdir $DIR_TO_CREATE
     done
     ```

Note : the list of directories is the list you got as output from the logs from step "g" or from the SQL script output from the SQL editor.

Enhanced security

To enhance the security of your application against clickjacking, we **highly recommend** to add following **response_headers_to_add** to the **envoy.Override.yaml** file, **route_config** object configuration.

```
...
route_config:
  name: local_route

  response_headers_to_add:
    - header:
        key: "X-XSS-Protection"
        value: "1"
      append: true
    - header:
        key: "X-Content-Type-Options"
        value: "nosniff"
      append: true
    - header:
        key: "Strict-Transport-Security"
        value: "max-age=3600; includeSubDomains"
      append: true
    - header:
        key: "Content-Security-Policy"
        value: "frame-ancestors 'self' teams.microsoft.com *.teams.microsoft.com *.skype.com"
      append: true
    - header:
        key: "X-Frame-Options"
        value: "DENY"
      append: true

  virtual_hosts:
  ...
```

4. Scaling

Although this is not part of the new release, we have provided you a scaling script in the repository to help you scale the Collaboard application in order to make optimal use of your environment and have better performances. The scaling script (called ***4_ScaleScript.sql***) is a SQL script that will update the scaling parameters in the SQL database.

**Installation steps ☑*

- in case you have access to the database via a SQL editor (e.g. SQL Server Management Studio), you can simple run the script into the editor.
- in case you do not have direct access to the SQL database, you can run the script using the DB-INIT method. This can be done through a configmap.

a. Check the database name in the 4_ScaleScript.sql script and update if needed.
b. Create a configmap called "cb-scaling-script" `kubectl create cm cb-scaling-script --from-file=ConfigurationMigration.sql=4_ScaleScript.sql`
c. Create a new job to mount the configmap with the SQL script and run it against the DB `kubectl create -f cb-job-scaling.yaml`
d. Check the logs output of the POD for a successful completion.

There are a few **breaking changes** present when updating to latest containers with "latest" tag.

For docker-compose:

1. Migration to Envoy

NGINX as main proxy in **cb_proxy** container has been replaced with **Envoy**. This brings support for horizontal rescaling of APIs and frontend container instances on the fly.

Steps to take:

- Remove the environmental variable **CB_OVERRIDE_NGINX** from the **cb_proxy** container.
- Remove file **nginx.Override.conf** from overrides folder you have mounted to your containers.
- Add file **envoy.Override.yaml** from Distribution repository. You can use the one here, uncomment the sections flagged with "With REDIS + Realtime" if your installation includes them.
- Add new variable **CB_OVERRIDE_ENVOY** with value **envoy.Override.yaml**.

CAUTION: In case you don't provide this optional override, default one will be used which includes cb_realtime container routing. ***If you don't use cb_realtime in your installation, default proxy configuration will cause up to 2 minutes delay in cb_proxy startup!!***

2. Database init/update

Originally the cb_database_init container contained SQL engine, which would directly create/access mounted database files. Now, the cb_database_init retained only the SQL commands execution ability and depends on running external/containerized SQL server.

Generic steps for all installations using cb_db container:

- Add environmental variable SA_PASSWORD to cb_db container with correct password.
- Install and start cb_db container before running the cb_database_init container.

DockerCompose steps:

1. Make sure you save your current installation configuration in different folder than cloned Distribution repository.
2. Pull latest version from Distribution repository master branch.
3. Navigate to folder **Distribution/DockerCompose/template**.
4. *Optional: Copy files to folder outside repository clone.*
5. **Replace the placeholders** and comment or uncomment respective "No REDIS" vs. "With REDIS + Realtime" sections in files:
   - appsettings.Override.json
   - env.Override.js
   - envoy.Override.yaml
   - ConfigurationMigration.sql
   - .env
   - docker-compose.yml
6. Make sure you are mounting **correct certificates** for **cb_proxy** and **cb_coturn** containers.
7. Run **update-and-run.*** script depending on your operating system with option "y" on confirmation prompts.

For Kubernetes/OpenShift:

1. Migration to Envoy

NGINX as main proxy in **cb_proxy** container has been replaced with **Envoy**. This brings support for horizontal rescaling of APIs and frontend container instances on the fly.

Steps to take:

- Save/backup your current configuration files (appsettings.Override.json, env.Override.js, ConfigurationMigration.sql), configmaps, service-manifests.
- All manifests have been updated in the git repository, so pull the latest version.
- Create new configmap :

  ```
  kubectl create cm cm-envoyconf --from-file=envoy.Override.yaml
  ```

- Delete the cb-proxy deployment :

  ```
  kubectl delete deployment cb-proxy
  ```

- Recreate the cb-proxy deployment :

  ```
  kubectl create deployment deployments/cb-proxy.yaml
  ```

  - To check:

    ```
    kubectl get po
    ```

  cb-proxy POD should be in **Running** STATUS

  - If not successful, check the logs with `kubectl logs <POD-NAME>` example : `kubectl logs cb-proxy-xxxzzs`

2. Services have been updated with different port numbers

- Delete updated services :

  ```
  kubectl delete svc api -n <YOURNAMESPACE>
  kubectl delete svc auth -n <YOURNAMESPACE>
  kubectl delete svc frontend -n <YOURNAMESPACE>
  kubectl delete svc licensing -n <YOURNAMESPACE>
  kubectl delete svc mftapi -n <YOURNAMESPACE>
  ```

- Recreate all services :

  ```
  kubectl create -f services/
  ```

3. Database init/update

Originally the cb_database_init container contained SQL engine, which would directly create/access mounted database files. Now, the cb_database_init retained only the SQL commands execution ability and depends on running external/containerized SQL server.

When you have a running Collaboard environment, you do not need to run this Database init/update containers. It is important at installation time or when you want to update and database parameters. In case you want to do an update, you no longer need to stop the database.