



ANARK COLLABORATE DEPLOYMENT

Contents

Platform Overview.....	6
Client Tier.....	6
Web Tier	6
Application Tier.....	6
Resource Tier	7
Minimum System Requirements	8
Client Devices	8
Desktop.....	8
Mobile.....	8
Browsers	9
Desktop.....	9
Mobile.....	9
Minimum System Hardware Requirements	10
Anark Collaborate Installation Instructions	11
Install System Prerequisites.....	11
NGINX	11
NodeJS	11
MongoDB.....	11
Redis	11
PM2.....	11

Anark Collaborate Deployment

Dotnet.....	12
Initial System Setup	12
Enabling Transport Layer Security (TLS)	15
Web Server	16
Application server.....	17
Database Server.....	19
Redis Server	20
Encryption at Rest.....	21
Block-Level Disk Encryption using LUKS.....	21
Targeted Encryption of a Data Disk	21
Key Management Using a Keyserver	22
Templates	23
Deploy Publish Worker	23
Linux Setup	24
Windows Setup.....	24
Configuring Anark Collaborate	26
PM2.json	26
Activity Report Worker	29
Download Worker	30
Upload Service	31
Upload Worker	32
Replication Service.....	33
Marking Service	35
Sandbox Worker	37
File Share Worker	39
Publish Worker Data.....	40
Publish Controller	41



Anark Collaborate Deployment

- Message Worker42
- Admin Portal – System preferences43
 - Content43
 - Content Properties Schema43
 - Shown Content Metadata44
 - Inherit Content Access from Item.....45
 - Access Control Web Service46
- Activities46
 - Shown Activity Metadata46
- Users46
 - Shown User Metadata46
 - User Sites46
 - Guest Users.....46
- Work Instructions47
 - Work Instruction Properties Schema.....47
 - Shown Work Instruction Metadata47
 - Work Instruction Execution Properties Schema47
 - Work Instruction Player Labels.....47
 - Work Instruction Access Control Web Service48
- Publishing48
 - Upload SeTtings48
 - Watermarks49
 - Data Markings.....51
- Notifications55
 - Enable Email Notifications55
 - SMTP Server.....56
 - Send From Email Address56



Anark Collaborate Deployment

Notification Email URLs	56
Search	57
Searchable and Sortable Properties	57
Activity	58
Content Item	59
Work Instruction	59
User	59
Group	59
Minimum Characters for Search	60
User Search Restrictions	60
Search Access Control Web Service	60
Miscellaneous	60
Item Access Caching	60
Job TTL	60
Logging	60
Define Help Files	61
Define Custom Help Links	61
Identity Management	63
Getting Started	63
Basic Set Up	64
Logging In	65
Attribute Mapping	67
Group Mapping	69
Request Signing	71
Logging Out	72
Administration	73
User Provisioning	75



Anark Collaborate Deployment

Role Management	79
Overview	79
Defining Custom Roles	79
Application Permissions	81
Troubleshooting and Log Files	83
High Availability	84
Web Server	84
Application Server	84
Database Server	84
Redis Server	85
Performance/Scaling Recommendations	88
Web Server	88
Application Server	88
Database Server	88
Security guidelines	90
Systems Integration	90
Authentication	90
Content Access Control (Authorization)	90
Accessing External Users and Groups	91
Licensing	92
Named User	92



PLATFORM OVERVIEW

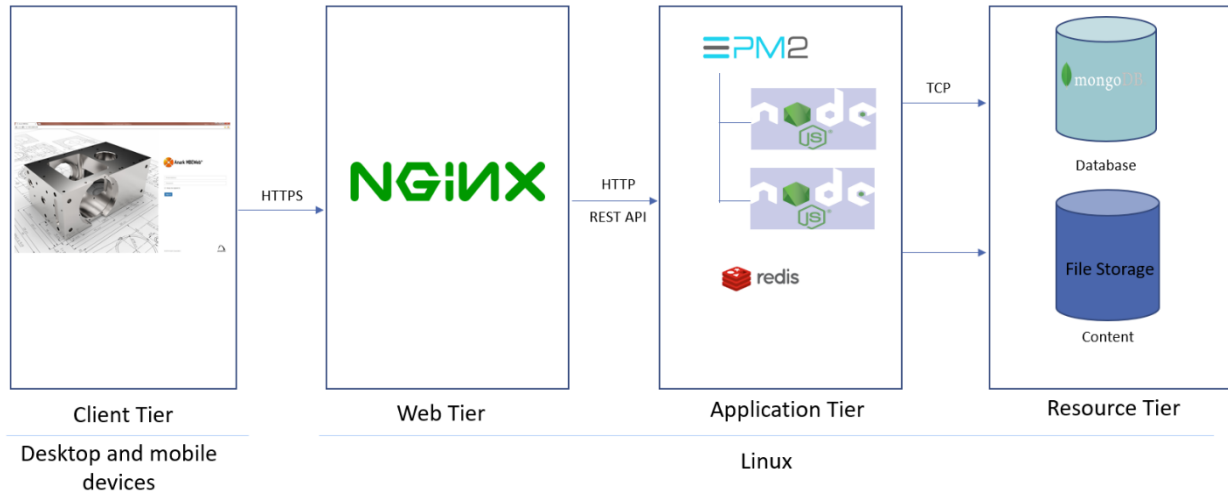


Figure 1: System Architecture Overview

The Anark Collaborate platform is built on scalable cloud technologies and consists of four tiers – client, web, application, and resource. The web, application, and resource tiers run on Linux, while knowledge workers across the enterprise can use a browser (client tier) running on desktop or mobile devices to access Anark Collaborate web applications.

CLIENT TIER

Authenticated users use the Anark Collaborate web applications to search, view and collaborate with the 2D or 3D product data that they are authorized for. Each user is assigned a role or multiple roles by the Anark Collaborate administrator. Depending on the assigned role(s), a user can access different apps and perform different actions in the system. The client tier uses HTTPS to connect to the Web tier.

WEB TIER

Anark Collaborate uses NGINX (version 1.22 or later) as the web server. NGINX is used to serve the static content (HTML and associated files for 3D and 2D content) and Anark Collaborate web applications. It provides SSL termination and acts as a reverse proxy to the application tier. It performs load balancing if there are multiple Node instances running in the application tier and performs health checks of the Node instances. NGINX delegates the authentication and authorization to the application tier and passes the REST API calls from client tier to the application tier.

APPLICATION TIER

Anark Collaborate uses Node.js as the application server. The application server hosts the Anark Collaborate REST API's. Multiple node instances can be run simultaneously for high performance and scalability. PM2 is used as the process manager for Node.js instances. Redis (version 6.0 or later) also runs in this tier and is used for high performance caching.



RESOURCE TIER

The Anark Collaborate resource tier consists of MongoDB database and file storage. Content metadata and collaboration related data are stored in MongoDB. File storage is used for storing content.



MINIMUM SYSTEM REQUIREMENTS

CLIENT DEVICES

Most modern mobile and desktop devices support the Anark Collaborate application.

Note: WebGL 1.0 support is a requirement for any client device to view 3D content in Anark Collaborate. For a list of unsupported client hardware and browser combinations, see the Khronos “BlacklistsAndWhitelists” page [here](#). To test whether your device and browser support WebGL, go to the WebGL Test site [here](#).

DESKTOP

OS	Version
OS X	12+
Windows	10

MOBILE

iOS	15+
Android	12+



BROWSERS

Most internet browsers for desktop and mobile should work with the Anark Collaborate application. The following lists are the most commonly used browsers and the earliest versions supported.

DESKTOP

Browser	Version
Microsoft Edge	80+
Google Chrome	80+
Safari	16+
Firefox	69.0.1+

MOBILE

iOS Safari	14+
Chrome for Android/Android Browser	80+



MINIMUM SYSTEM HARDWARE REQUIREMENTS

Anark Collaborate Server Infrastructure	Standard Deployment	Improved Deployment	Performance Deployment
Web Server + Application Server + File Server	64-bit RHEL 8.x box (VM) 6 cores 64 GB RAM HDD 8 TB, RAID 1, 15k RPM (adequate for app. 200,000 contents assuming 25 MB average)	64-bit RHEL 8.x box (VM) 6 cores 32 GB RAM HDD 5 TB, RAID 1, 15k RPM (adequate for app. 200,000 contents assuming 25 MB average)	64-bit RHEL 8.x box (VM) 8 cores 64 GB RAM SSD - 5 TB (adequate for app. 200,000 contents assuming 25 MB average)
Database Server	Database installed together on server above	64-bit RHEL 8.x box (VM) 4 cores 64 GB RAM (more RAM will be better, to avoid the need for sharding in the future) HDD 1 TB, RAID 1, 15k RPM	64-bit RHEL 8.x box (VM) 6 cores 64 GB RAM (more RAM will be better, to avoid the need for sharding in the future) SSD - 1 TB
Replication/Failover Servers	If there are multiple sites that share all data (files, database, etc.), then failover will be accomplished by temporarily redirecting to another site. If multiple sites have independent data (files, database, etc.) separate failover server(s) will be required.	If there are multiple sites that share all data (files, database, etc.), then failover will be accomplished by temporarily redirecting to another site. If multiple sites have independent data (files, database, etc.) separate failover server(s) will be required.	If there are multiple sites that share all data (files, database, etc.), then failover will be accomplished by temporarily redirecting to another site. If multiple sites have independent data (files, database, etc.) separate failover server(s) will be required.

Note: Backup for file server and database not specified above but is required.



ANARK COLLABORATE INSTALLATION INSTRUCTIONS

INSTALL SYSTEM PREREQUISITES

Install the required server components on RHEL 8.x using the yum package manager.

NGINX

NGINX must be installed at version 1.24 or newer. Additionally, NGINX dynamic module NJS ([njs scripting language \(nginx.org\)](https://nginx.org/en/docs/http/ngx_http_js_module.html)) must also be installed. This can be done by creating the file named `/etc/yum.repos.d/nginx.repo` with the following contents:

```
[nginx-stable]
name=nginx stable repo
baseurl=http://nginx.org/packages/centos/$releasever/$basearch/
gpgcheck=1
enabled=1
gpgkey=https://nginx.org/keys/nginx_signing.key
module_hotfixes=true
```

and then running the command “`yum install nginx-module-njs`”. Accept the option to install NJS and NGINX.

NODEJS

NodeJS must be installed at version 20 or later.

For manual installation in RedHat systems:

```
dnf module enable nodejs:20
dnf install nodejs
```

MONGODB

MongoDB must be of version 6.0. Follow these instructions to install MongoDB on Redhat: [Install MongoDB Community Edition on Red Hat or CentOS — MongoDB Manual](#)

REDIS

Redis must be installed at version 6.2.7 or newer.

For RedHat systems:

```
dnf module enable redis:6
dnf install redis
systemctl enable redis
```

PM2

PM2 (<http://pm2.io/>) must be of version 10.5.0 or newer



Anark Collaborate Deployment

```
npm install pm2 -g
```

DOTNET

Dotnet must be installed at version 8.x.

```
dnf install dotnet-sdk-8.0 -y
```

INITIAL SYSTEM SETUP

- 1) Use an existing user or create a new user with any username (Ex: *node-collab*). Skip steps 2 and 3 if using an existing user account.

```
useradd -m node-collab
```

- 2) Give the *node-collab* user a strong and unique password using the *passwd* command.
- 3) Create a new group *collab* and add *node-collab* to this group.

```
groupadd collab  
usermod -a -G collab node-collab
```

- 4) Enable redis to start on boot:

```
systemctl enable redis
```

- 5) Create an admin user in MongoDB.

a. Go to mongo shell (type “mongosh” in shell) and enter the below commands. You can use a different username (user) and password (pwd).

```
use admin  
db.createUser (  
  {  
    user: "admin", pwd: "admin",  
    roles: [ {role: "userAdminAnyDatabase", db: "admin"} ]  
  }  
)  
exit
```

- 6) Modify MongoDB config file.

a. Open the MongoDB config file, “/etc/mongod.conf” and set the “authorization” setting under “security” to “enabled.”



Anark Collaborate Deployment

- b. Then change the `unixDomainSocket` from `true` to `false`.

```
unixDomainSocket:
  enabled: false
...
security:
  authorization: enabled
```

- 7) Restart MongoDB and Redis using the below commands.

```
systemctl restart mongod
systemctl restart redis
```

- 8) Create a new MongoDB user for Anark Collaborate. Go to mongo shell (type “`mongosh`” in shell) and enter the below commands.

```
use admin
db.auth("admin", "admin" ) // Use the username and password used to create the admin user
use anarkmbedev

db.createUser (
  {
    user: "collab", pwd: "collab",
    roles: [ {role: "readWrite", db: "anarkmbedev"} ]
  }
)
exit
```

- 9) Download `collaborate.tar.gz`, and run

```
tar -xzf - --strip-components=2 -C /usr/local -f collaborate.tar.gz
```

to extract the Anark Collaborate system package. This should extract the files to the “`/usr/local/collab`” directory.

- 10) Navigate to the “`/usr/local/collab/node_modules`” directory. Here you find three folders containing node packages compiled against different versions of NodeJS. Copy the contents of the folder that corresponds to the NodeJS version you are running, into the parent directory “`/usr/local/collab/node_modules`”.

- 11) Make the “`node-collab`” user the owner of the below folders and files. If the file or directory does not exist, create it and then apply the permission.

```
chown -R node-collab:collab <folder name> or chown node-collab:collab <file name>
```

- `/usr/local/collab`
- `/var/log/collab.log`
- `/var/log/nginx/error.log`



Anark Collaborate Deployment

- /var/tmp/client_body
- /var/cache

If the content store is not located in “/usr/local/collab/contentstore”, modify the permissions of that folder as well so that “node-collab” user can read/write to that folder.

12) Download nginx.tar.gz, and run

```
tar -xz --strip-components=1 -C /etc -f nginx.tar.gz
```

to extract the nginx configuration files. This should extract the files to the “/etc/nginx” directory.

13) Open the NGINX config file (/etc/nginx/nginx.conf) and edit the port numbers (1337 and 1338) if these are already used. NGINX worker processes are run under “node-collab” user. If you are using a different user account, update the first line (“user node-collab collab”) of the config line with the new user information (Ex: user <userid> <group>).

Additionally, if setup requires overriding variables such as content store location, cache location, or error log location, or if TLS needs to be enabled, this can be done in nginx.conf.

14) Enable access to ports by running. NOTE: You may need to install *firewalld* first.

```
semanage port -a -t http_port_t -p tcp <1337 or new port #>  
semanage port -a -t http_port_t -p tcp <1338 or new port #>  
firewall-cmd --zone=public --add-port=80/tcp --permanent  
firewall-cmd --zone=public --add-port=443/tcp --permanent
```

15) Restart NGINX

```
systemctl restart nginx
```

16) Open the “/usr/local/collab/pm2.json” file and modify the port numbers (1337 and 1338) if you changed the ports (Step 13).

17) Create PM2 startup script by typing in the below command. This creates the necessary script to start Anark Collaborate Node processes at boot.

```
pm2 startup systemd -u node-collab --hp /home/node-collab
```

18) After logging in as “node-collab” user, save the process list using the below commands.

```
cd /usr/local/collab  
pm2 start pm2.json
```



```
pm2 save
```

- 19) Modify kernel settings on production server (QA server if needed) to optimize performance. Please refer to the below links for recommended values for `sys.fs.file-max`, `kernel.pid_max`, `kernel.threads-max` and `net.core.somaxconn`.

<https://docs.mongodb.com/manual/reference/ulimit/>

<https://docs.mongodb.com/manual/administration/production-checklist-operations/>

<https://www.nginx.com/blog/tuning-nginx/>

- 20) Reboot and verify that all the server components (NGINX, Node.js, Redis, and MongoDB) started up correctly.
- 21) Set up the admin user for Anark Collaborate. Go to `"/usr/local/collab"`, open `"scripts/addadminuser.js"`, change the password and email (optional), and save the file. Then run

```
node scripts/addadminuser.js
```

This should create the Anark Collaborate admin user and you should see a status whether the user was successfully created.

- 22) You should be able to access Anark Collaborate now and login to the admin portal to manage users.

- 23) Download `fonts.tar.gz` and extract the fonts package to the `"/usr/share/fonts/msttcore"` directory:

```
tar -xzf fonts.tar.gz
```

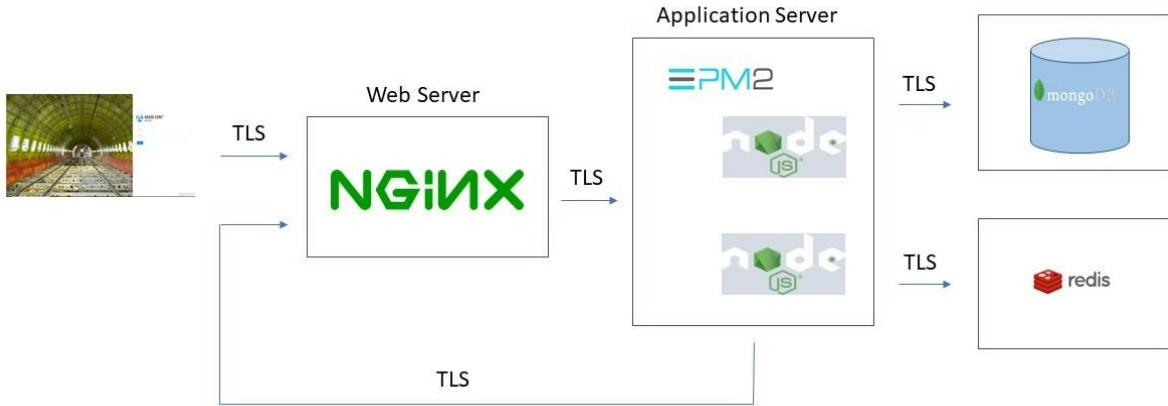
- 24) If intending to run SandboxWorker for Data Markings you must additionally run the following command:

```
dnf install python3 make gcc gcc-c++ zlib-devel brotli-devel openssl-devel
```

ENABLING TRANSPORT LAYER SECURITY (TLS)

Anark Collaborate server components can be configured to use one-way or mutual (two-way) TLS. You can set it up for one-way TLS, where the server will present the certificate to the client to verify its identity or use mutual TLS where both server and client's identity can be verified. As shown in the diagram below, TLS can be set up between every layer of Anark Collaborate including the browser to web server, web server to application server, application server to Mongo database server, and application server to Redis server. Service to service communication is routed through the web server and it can be configured to use TLS as well.





WEB SERVER

Follow the documentation below for setting up One-Way TLS or Mutual TLS. After making nginx conf changes, restart the NGINX server using the below command.

```
systemctl restart nginx
```

ONE-WAY TLS

Once a signed certificate has been obtained, make the following changes to your “nginx.conf” configuration file to enable TLS for your Web server. The Web server will present this certificate to clients to verify its identity.

- Copy the certificate (.crt) file to “/etc/pki/tls/certs” directory.
- Copy the key file (.key) to the “/etc/pki/tls/private” directory.
- Comment out the existing line
listen 80 default_server;
- Add the below lines in the “nginx.conf” file (“server” block) to enable SSL.

```
# Redirect http request to https
server {
    listen 80 default_server;
    return 301 https://$host$request_uri;
}

listen 443 ssl http2 default_server;
ssl_certificate /etc/pki/tls/certs/nginx.crt;
ssl_certificate_key /etc/pki/tls/private/nginx.key;
ssl_session_cache shared:SSL:10m;
ssl_session_timeout 10m;
ssl_protocols TLSv1.2 TLSv1.3;
ssl_ciphers HIGH:!aNULL:!MD5;
ssl_prefer_server_ciphers on;
add_header Strict-Transport-Security "max-age=31536000" always;
```

Please refer to the NGINX doc, link below, for more information about the above SSL directives.



https://nginx.org/en/docs/http/nginx_http_ssl_module.html#ssl_certificate

MUTUAL TLS

To enable mutual TLS, add the below lines to the “server” block in the “nginx.conf” file.

```
ssl_client_certificate    /etc/pki/tls/certs/ca.crt;  
ssl_verify_client        optional;
```

“ssl_client_certificate” directive can be used to specify the file with trusted CA certificates to verify client certificates. The “ssl_verify_client” directive can be used to enable the verification of client certificates. The value “optional” is one of the possible settings, in this case, where the server will request the client certificate and verify it, only if the certificate is present. It can be set to “on” to verify the client certificate always. Once client certificate authentication is enabled, the browser will need to send the client certificate when accessing Anark Collaborate.

If mutual TLS is enabled in the application server, the web server needs to be configured to pass the client certificate. Add the below lines in the server block.

```
proxy_ssl_certificate    /etc/pki/tls/certs/client.crt;  
proxy_ssl_certificate_key /etc/pki/tls/private/client.key;  
proxy_ssl_protocols     TLSv1.2 TLSv1.3;  
proxy_ssl_ciphers       HIGH:!aNULL:!MD5;  
proxy_ssl_trusted_certificate /etc/pki/tls/certs/trusted_ca_cert.crt;  
proxy_ssl_verify        on;  
proxy_ssl_verify_depth  2;  
proxy_ssl_session_reuse on;  
proxy_ssl_name          <hostname>
```

Please refer to the NGINX docs (link below) for more information about the proxy_ssl directives.

http://nginx.org/en/docs/http/nginx_http_proxy_module.html

APPLICATION SERVER

If the application server is configured for TLS, all the “proxy_pass” directives in the nginx conf file need to be modified to use “https” protocol. Service to service API call is routed through the web server. If web server is set up for TLS, modify all the URLs (AUTH_SERVICE_URL, CONTENT_SERVICE_URL, and CORE_SERVICE_URL) in pm2.json to use https instead of http.

NODE_EXTRA_CA_CERTS setting should be used to specify additional root CA certificates if using a custom certificate authority, otherwise Node.js runtime will not be able to validate certificates. More information on this setting can be found in Node.js documentation (https://nodejs.org/api/cli.html#node_extra_ca_certfile). Certificate should be in PEM format and multiple certificates can be bundled together. If the web server is configured to use TLS and a custom certificate authority is used to sign the TLS certificate, add this setting to any service in the pm2.json file that has an URL setting (CORE_SERVICE_URL, CONTENT_SERVICE_URL, or AUTH_SERVICE_URL). Also, this setting is required if two-way TLS is set up for the application server and a custom certificate authority is used to sign the client certificate (web server), add it to any service that has ENABLEMUTUALTLS setting set to true.



ONE-WAY TLS

To configure one-way TLS for Anark Collaborate microservices, configure the below settings in the PM2.json file for the Core service, Upload service, and Replication service. Since you might be running more than one instance of a service, be sure to configure all service instances. This configuration will allow the web server to verify the identity of the server that runs the microservice.

If you are using the shipped PM2.json, modify the below settings in **“collab-0”**, **“collab-1”**, **uploadservice-0**, and **replicationservice-0** under the **“env”** property, as shown in the below example.

```
“ENABLETLS”: true,  
“TLCERTPATH”: “/etc/pki/tls/certs/server.crt”,  
“TLSKEYPATH”: “/etc/pki/tls/private/server.key”,  
“TLSKEYENCRYPTIONPASSWORD”: “dac8a4aed5ae397c0eaea8b45a03c4903b...”
```

ENABLETLS

Use this setting to enable or disable TLS.

TLCERTPATH

Use this setting to specify the path to the server certificate that needs to be used. Certificate should be in PEM format.

TLSKEYPATH

Use this setting to specify the path to the private key that needs to be used. Private key should be in PEM format. Private keys can optionally be encrypted.

TLSKEYENCRYPTIONPASSWORD

Use this setting to specify the password if the private key is encrypted. Password can optionally be encrypted, Anark Collaborate provides a script, *encrypt.js*, to encrypt passwords. The PM2.json section below has more information on how to use the utility.

MUTUAL TLS

If mutual TLS needs to be supported, add the below setting to Core, Upload, and Replication service. Modify the below setting in **“collab-0”**, **“collab-1”**, **uploadservice-0**, and **replicationservice-0** under the **“env”** property.

```
“ENABLEMUTUALTLS”: true
```

ENABLEMUTUALTLS

Use this setting to enable or disable mutual TLS, enabling this setting will cause the server to request a certificate from the clients when they connect.



If mutual TLS is enabled, when the web server talks to the application server running microservices it needs to pass a client certificate to verify its identity. The [MUTUAL TLS](#) section above has more information on how to configure the web server to pass client certificate.

If mutual TLS is enabled on the web server, several microservices need to be configured using the below settings to present the client certificate to the web server. Modify the below settings in the **uploadservice-0**, **uploadworker-0**, **downloadworker-0**, **replicationservice-0**, **publishcontroller-0**, **markingservice-image**, **markingservice-pdf**, **messageworker-0**, **activityreportwork-0**, and **fileshareworker-0** under the “env” property.

```
“TLSCLIENTCERTPATH”: “/etc/pki/tls/certs/client.crt”,  
“TLSCLIENTKEYPATH”: “/etc/pki/tls/private/client.key”,  
“TLSCLIENTKEYENCRYPTIONPASSWORD”: “ad894aed5ae397c0eaea8b45a03c4903b...”
```

TLSCLIENTCERTPATH

Use this setting to specify the path to the client certificate that needs to be used. Certificate should be in PEM format. This client certificate will be used to connect to the web server.

TLSCLIENTKEYPATH

Use this setting to specify the path to the private key that needs to be used. Private key should be in PEM format. Private keys can optionally be encrypted.

TLSCLIENTKEYENCRYPTIONPASSWORD

Use this setting to specify the password if the private key is encrypted. Password can optionally be encrypted, Anark Collaborate provides a script, *encrypt.js*, to encrypt passwords. The PM2.json section below has more information on how to use the utility.

DATABASE SERVER

MongoDB server can be configured to support one-way or mutual TLS configuration. Please refer to the MongoDB documentation (link below) for more information.

<https://docs.mongodb.com/manual/tutorial/configure-ssl/>

ONE-WAY TLS

To connect to a Mongo DB server that is configured for one-way TLS, use the below connection string.

```
mongodb://localhost:27017/anarkmbedev?tls=true&tlsCAFILE=<locationofcafile>
```

tlsCAFile is used to specify the location of a file that contains the root certificate chain from the Certificate Authority. This is used to validate the server certificate presented by the MongoDB server.

MUTUAL TLS

To connect to a Mongo DB server that is configured for mutual TLS, use the below connection string.



```
mongodb://localhost:27017/anarkmbedev?tls=true&tlsCertificateKeyFile=<pathtokeyfile>&
tlsCAFile=<locationofcafile>&tlsCertificateKeyFilePassword=<password>
```

tlsCertificateKeyFile is the location of the client's Privacy Enhanced Mail (PEM) key and certificate.

tlsCertificateKeyFilePassword is an optional parameter to specify the password to decrypt the key file. Password can optionally be encrypted, Anark Collaborate provides a script, *encrypt.js*, to encrypt passwords. The PM2.json section below has more information on how to use the utility.

tlsCAFile is used to specify the location of a file that contains the root certificate chain from the Certificate Authority. This is used to validate the server certificate presented by MongoDB server.

REDIS SERVER

Redis server can be configured to support one-way or mutual TLS configuration. Please refer to the Redis documentation (link below) for more information.

<https://redis.io/topics/encryption>

ONE-WAY TLS

To connect to Redis server that is configured for one-way TLS, use the below connection string.

```
"redis://localhost:6379?tls=true"
```

MUTUAL TLS

To connect to Redis server that is configured for mutual TLS, use the below connection string.

```
redis://localhost:6379?tls=true&tlsCertificateFile=<pathtocertfile>&tlsCertificateKeyFile=<pathtokeyfile>&tl
sCertificateKeyFilePassword=<password>
```

tlsCertificateFile is the location of the client certificate.

tlsCertificateKeyFile is the location of the client's private key file.

tlsCertificateKeyFilePassword is an optional parameter to specify the password to decrypt the key file. Password can optionally be encrypted, Anark Collaborate provides a script, *encrypt.js*, to encrypt passwords. The PM2.json section below has more information on how to use the utility.



ENCRYPTION AT REST

Anark recommends the following practices for encryption of data at rest:

1. Deploy Anark Collaborate VMs using Linux Unified Key Setup (LUKS) block-level encryption
 - Best overall option for performance, security, and flexibility
2. Use full-disk encryption on a separate data disk instead of the root disk
 - Simplifies deployment, configuration, and troubleshooting
3. Store and manage the encryption key using an external keyserver
 - Allows for automation to improve availability and security
4. For database encryption, use MongoDB Enterprise which provides [encryption at rest](#) feature

BLOCK-LEVEL DISK ENCRYPTION USING LUKS

Anark recommends block-level disk encryption using LUKS ([Encrypting block devices using LUKS](#)) to ensure confidentiality of Anark Collaborate data at rest. As part of the Linux kernel, LUKS is the standard way to achieve block-level disk encryption on Linux, and enjoys many benefits including:

- Minimal performance overhead for encryption and decryption.
- Extensive documentation and tooling.
- Integration with other parts of the Linux kernel, including logical volume management (LVM), which is used by Anark Collaborate.
- No special hardware requirements.

LUKS encrypts entire disks, partitions, or logical volumes (the combination of which shall henceforth be referred to as "volumes"). Before mounting an encrypted volume, a symmetric block-level encryption key must be provided to LUKS, which then mounts the volume "transparently": this means that the volume appears as unencrypted to the operating system but is still encrypted on disk. Access control must be used to prevent unauthorized processes from reading the mounted volume. Also note that data read from a LUKS-encrypted volume will be unencrypted in memory.

TARGETED ENCRYPTION OF A DATA DISK

Anark recommends encrypting a separate disk specifically for Anark Collaborate data (including separate volumes for the Content Store, MongoDB, and Redis) and keeping the root disk unencrypted, so no password will be required when booting the VM. The main advantage of password-less boots is that Anark Collaborate can start without human intervention, a crucial prerequisite for a high-availability deployment.

Note: Although it is technically possible to automatically decrypt a LUKS-encrypted root disk at startup using policy-based decryption (PBD), the required infrastructure and maintenance thereof is a heavy burden. See [this Red Hat guide for reference](#).

As for decryption of the data disk, Anark recommends automating this process as part of system initialization. We provide a suggested implementation in the upcoming [Key Management Using a Keyserver](#) section.



Another advantage offered by limiting LUKS encryption to the data disk is that initial configuration and troubleshooting of Anark Collaborate may be simplified by deferring security setup until everything else has been tested and validated. When you are ready, LUKS encryption can then be rolled out incrementally. For example, during manual setup of Anark Collaborate, you may decide to follow a progression like this:

1. Start with an unencrypted data disk with separate volumes for the Content Store, MongoDB, and Redis.
2. Perform a few smoke tests against the data disk from Anark Collaborate, writing dummy data that needs no encryption.
3. Encrypt the entire data disk using LUKS.
4. Manually mount each logical volume (interactively supplying the encryption key to LUKS) to validate the mount operations.
5. Perform a few more smoke tests against the data disk from Anark Collaborate, ensuring that the data disk and its logical volumes are still working under LUKS.
6. Automate the process of unlocking and mounting the data disk, and test that automation in your running VM.
7. Reboot the system to test your automation at boot time.

KEY MANAGEMENT USING A KEYSERVER

Anark recommends generating, storing, and managing the block-level encryption key on an external keyserver. Doing so could offer the following advantages:

- The possibility for secure highly available deployments. (Programmatically obtaining the block-level symmetric encryption key from a keyserver can be used to automate disk decryption at startup, resulting in unattended boot sequences without sacrificing security. As stated previously in this document, unattended boot sequences are a prerequisite for high availability.) One possible implementation would be to use an init script to:
 1. Authenticate to the keyserver using credentials assigned to the VM through some external service.
 2. Retrieve the block-level symmetric encryption key from the keyserver.
 3. Unlock the encrypted disk using that key.
 4. Mount the disk.
- Improved security through automated key rotation. One possible implementation would use a cron job (scheduled for maintenance windows) to run a script to:
 1. Authenticate to the keyserver using credentials assigned to the VM through some external service.
 2. Initiate a rotation of the block-level symmetric encryption key on the keyserver.
 3. Obtain the new key from the keyserver.
 4. Rotate the key against the LUKS-encrypted data disk (using LUKS2, this might not even require unmounting the disk first).
 5. If step 4 fails, rollback the key rotation in the keyserver.

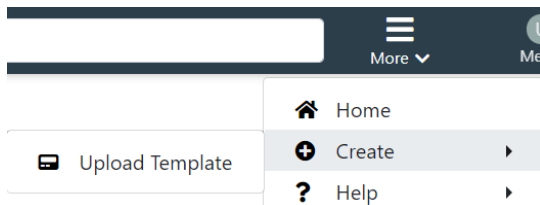


Regardless of how the automation is implemented, the credentials used by the VM to authenticate to the keyserver should **never** be stored on the VM's disk. Rather, some attribute of the VM unique to your specific operating environment should be used to identify it and associate credentials with it.

TEMPLATES

Publishing content from Anark Workstation or Anark Publish requires a user uploaded template. Users can upload custom templates with more advanced capabilities by following the steps below. Additionally, Anark Collaborate comes with an out-of-the-box template `ToolbarPanelViewer.zip` that can be uploaded and used.

Once the initial system is configured and running, create a user with the “content author” role. Login to the system as this user and upload a template used by your organization if necessary. You can access this feature by selecting the “Upload Template” button from the navbar “plus” icon.



Fill out the below fields and click “Upload”.

A screenshot of the 'Upload Template' dialog box. The dialog has a dark header with the title 'Upload Template' and a close button (X). Below the header, there are three input fields: 'Name (Required)' with a text input field, 'Description' with a text area, and 'File' with a 'Choose File' button and the text 'No file chosen'. At the bottom of the dialog, there are two buttons: 'Cancel' and 'Upload'.

DEPLOY PUBLISH WORKER

Publish worker, a component of publishing services, needs to be deployed for browser-based publishing via Anark Collaborate UI (My Files, Activity). It requires Windows Server 2019 and .NET 4.7.2. Publish worker needs to communicate with Redis server used for Anark Collaborate deployment, if there are any connection issues, ensure that Redis config “bind” setting is correctly set to allow external connections and firewall is not preventing access. It requires setup on both Linux (where the rest of the Anark Collaborate components are running) and Windows server.



LINUX SETUP

- 1) Create the directory that will be used for “jobdatafolder” staging share. Make the user created in [Initial System Setup \(Step 1\)](#) the owner of the folder.

```
chown -R node-collab:collab <folder name> or chown node-collab:collab <file name>
```

- 2) Staging share for “jobdatafolder” needs to be accessible with read and write permissions on Windows. To set up the share using Samba:

- a. Install the Samba package.

```
dnf install samba -y
```

- b. Modify the configuration file (smb.conf) to configure the shared directory. Typically, it is located under /etc/samba folder.

```
[jobdata]
comment = Jobdata share
path = <jobdata location>
read only = No
```

- c. Use the “smbpasswd” command to add the user used to run Anark Collaborate to smbpasswd file.

```
smbpasswd -a node-collab
```

- d. Configure SELinux on the share so that Samba service has proper permissions.

```
semanage fcontext -a -t samba_share_t <jobdata location>
restorecon -R -v <jobdata location>
```

- e. Enable and start Samba services.

```
systemctl enable smb nmb
systemctl start smb
systemctl start nmb
```

- f. Configure firewall to allow Samba traffic.

```
firewall-cmd --permanent --add-service=samba
firewall-cmd --reload
```

More details on setting up the Samba share can be found in the below links.

<https://www.redhat.com/sysadmin/getting-started-samba>

<https://www.redhat.com/sysadmin/samba-windows-linux>

WINDOWS SETUP

- 1) Extract the contents of the package (AnarkCollaboratePublishWorker.zip) to a folder on the server.
- 2) Install the Visual C++ Redistributable for Visual Studio 2019. Installer for the redistributable (vc_redist.x64.exe) is included in the package (AnarkCollaboratePublishWorker.zip).



- 3) Install [Node.js 20](#).
- 4) Install PM2 as a Windows service using pm2-installer. The package for installing pm2-installer can be found in the package (AnarkCollaboratePublishWorker.zip), under “pm2-installer” folder. Typically for a fresh install, you will run the following commands in an elevated PowerShell terminal, “npm run configure”, “npm run configure-policy” and “npm run setup”. Offline installation is supported, if the machine is connected to the internet, it will pull the packages from the npm registry. Please refer to the help file (“pm2-installer-help.pdf”) for more detailed information on installation.
- 5) Create a local user for Samba sharing, use the same login and password as the user account that was setup on Linux for Samba sharing.
- 6) Modify permissions for “services” and “home” folders located under “C:\ProgramData\pm2” folder – give “Full Control” permission to the local user account used for Samba sharing.
- 7) PM2 windows service should run using the user account that has access to Samba share. To change the user account, open the Services app, locate “PM2” service, right click and select Properties to open the “Properties” dialog. Go to “Log on” tab, modify the account username and password, and click “Apply”. Restart the service by right clicking on “PM2” service and selecting “Restart”.
- 8) Login using the local user and test the “jobdatafolder” share access, use UNC path to access the share. Configure the settings in pm2.json as described below. Once configuration is completed, start the publish worker and save the process list using pm2 save command.

```
cd <INSTALLFOLDER>\Anark\Core\Workers\PublishWorker
pm2 start pm2.json
pm2 save
```

CONFIGURATION

Configure the settings in PM2.json file located in <INSTALLFOLDER>\Anark\Core\Workers\PublishWorker folder. You can run more than one instance of publish worker if needed.

PORT

By default, Publish Worker will run on port 5000. To use a different port, change this option.

REDIS

This setting specifies the location where Redis is running.

```
“REDIS”: “redis://<server>:6379”
```

REDIS USERNAME

This setting is identical to the **REDIS_USERNAME** setting in Anark Collaborate. Please refer to it [below](#).

REDIS PASSWORD

This setting is identical to the **REDISPASSWORD** setting in Anark Collaborate. Please refer to it [below](#).



ENCRYPTION SECRET FILEPATH

This setting is identical to the **ENCRYPTION_SECRET_FILEPATH** setting in Anark Collaborate. Please refer to it [below](#).

JOB DATA FOLDER

This setting specifies where files associated with a job will be temporarily stored while the job processes. Specify the staging share location using UNC path.

```
"JOBDATAFOLDER": "\\172.22.0.154\jobdata"
```

SITE

If Anark Collaborate is deployed in multiple sites, specify the name - must be unique - to identify the site.

CLOUD DEPLOYMENT

This setting should be set to false.

CONFIGURING ANARK COLLABORATE

Anark Collaborate must be configured to use specific ports to function properly. Anark Collaborate uses port 443 (HTTPS) or 80 (HTTP) for the web server, 1337 and 1338 for the application server, 27017 for MongoDB, 6379 for Redis, 26379 for Redis Sentinel (if Redis is configured for High Availability). If these ports are already in use, the system can be configured to use different ports. Check the firewall to verify that the port used by the web server has access through the firewall.

Anark Collaborate is configured using the pm2.json file and through the Admin portal. Most of the configuration preferences in the Admin portal can be modified and the changes will be immediately available.

PM2.JSON

First, in the pm2.json file make the following changes in both "collab-0" and "collab-1" under the "env" property, following the below example.

```
"PORT": 1337,  
"NODE_ENV": "production",  
"DB": "mongodb://localhost:27017/anarkmbedev",  
"DBUSER": "collabusername",  
"DBPASSWORD": "dac8a4aed5ae397c0eaea8b45a03c4903b...",  
"REDIS": "redis://localhost:6379",  
"REDISPASSWORD": "ed8e6da3bb6969f09a103ce742f6c6a0...",  
"FSROOT": "/usr/local/collab/contentstore",  
"FSSYNURLS": "",  
"FSSYNCTIMEOUT": 0,  
"SESSIONTIMEOUT": 7200,  
"JOBDATAFOLDER": "/var/cache/collab/jobdata",
```



```
"ENCRYPTION_SECRET_FILEPATH": "/etc/secrets/mbeweb-secret.key"
```

Properties **DBPASSWORD** and **REDISPASSWORD** can optionally be encrypted, Anark Collaborate provides a script, `encrypt.js`, to encrypt and decrypt passwords. It is recommended that the encryption secret is stored in a file. When using the `encrypt.js` script, provide the `(-f)` argument. If providing the secret in command line `(-s)`, be careful that the secret does not contain any special characters that the shell may interpret incorrectly.

`Encrypt.js` can be found in `/usr/local/collab/scripts/` and takes the following arguments:

```
-p / --password <your password to be encrypted>  
-d / --decrypt <your encrypted password to decrypt>  
-f / --file <encryption secret file path>  
-s / --secret <encryption secret string>
```

Example:

```
node scripts/encrypt.js -p testpassword -f /etc/mbeweb-secret.key
```

ANARK COLLABORATE

`PM2.json` contains two Node servers for running Anark Collaborate with the names "collab-0" and "collab-1". Each Node server should have almost identical environmental settings (with a different port for each). The following properties are available to configure in these Anark Collaborate servers.

PORT

By default, Anark Collaborate application server runs two Node servers, it uses **PORT 1337** and **PORT 1338**. If these ports are not available or if you want to use a different port, use this setting to specify the port.

NODE ENVIRONMENT

The **NODE_ENV** property should be set to "production" in Production servers for performance and to disable verbose logging. Set this to "development" (Dev and QA servers) to help with debugging. Additionally, if either **NODE_ENV** is set to "production" or **ENABLETLS** is set to true, Anark Collaborate will use Secure Cookies.

DATABASE

The **DB** property specifies the MongoDB connection string. In general, it should take the following form. The High Availability section below has details on how to configure MongoDB replica sets.

```
mongodb://<host>:<port>/<database >
```

DATABASE USER

The **DBUSER** property specifies the username to use when connecting to MongoDB.



DATABASE PASSWORD

The **DBPASSWORD** property specifies the encrypted password associated with **DBUSER** which is required to authenticate with MongoDB. When specifying **DBPASSWORD**, the secret used to encrypt the password must be provided via the **ENCRYPTION_SECRET_FILEPATH** property.

ENCRYPTION SECRET FILEPATH

ENCRYPTION_SECRET_FILEPATH takes a file path and file name that Anark Collaborate will use to obtain the encryption secret.

For example:

```
"ENCRYPTION_SECRET_FILEPATH": "/etc/secrets/mbeweb-secret.key",
```

REDIS

Specify the Redis or Redis Sentinel connection string using **REDIS** property. Redis Sentinel is used for high availability. If the system is not set up for high availability, then this property should take the following forms:

```
"REDIS": "redis://<host>:<port>",
```

If the system is setup for high availability, then this property should resemble this form instead:

```
"REDIS": "<host_1>:<port>,<host_2>:<port>, <host_3>:<port>",
```

FILE SYSTEM ROOT

Use **FSROOT** to specify the content store for Anark Collaborate.

```
"FSROOT": "/usr/local/collab/contentstore"
```

SESSION TIMEOUT

SESSIOINTIMEOUT specifies when to automatically sign out users who are inactive. By default, it is set to 2 hours – or 7200 seconds.

REDIS USERNAME

If Redis is configured for authenticating with a username, then the **REDIS_USERNAME** property specifies the username used to connect to Redis.

REDIS PASSWORD

The **REDISPASSWORD** property specifies the password, encrypted or not, used to connect to Redis. If the password is encrypted, the secret used to encrypt the password must be provided via the **ENCRYPTION_SECRET_FILEPATH** property.



JOB DATA FOLDER

This setting specifies where files associated with a job will be temporarily stored while the job processes.

```
"JOBDATAFOLDER": "/var/cache/collob/jobdata"
```

SITE

If Anark Collaborate is deployed in multiple sites, specify the name - must be unique - to identify the site.

ENABLE REPLICATION

To enable or disable content replication between multiple sites, specify true or false for this setting.

```
"REPLICATIONENABLED": true
```

CONTENT SERVICE URL

Specify the **CONTENT_SERVICE_URL** to validate content in upload requests.

```
"CONTENT_SERVICE_URL": http://localhost:80
```

ACTIVITY REPORT WORKER

To use the Activity Report worker, a separate pod must be configured in PM2.json. This pod is named "activityreportworker-0". The available environmental settings are detailed below.

PORT

By default, Activity Report Worker will run on port 8091. To use a different port, change this option.

REDIS

This setting specifies the location where Redis is running.

```
"REDIS": "redis://<server>:6379"
```

REDIS USERNAME

If Redis is configured for authenticating with a username, then the **REDIS_USERNAME** property specifies the username used to connect to Redis.

REDIS PASSWORD



Anark Collaborate Deployment

This setting is identical to the **REDISPASSWORD** setting in Anark Collaborate. Please refer to it above.

ENCRYPTION SECRET FILEPATH

This setting is identical to the **ENCRYPTION_SECRET_FILEPATH** setting in Anark Collaborate. Please refer to it above.

JOB DATA FOLDER

This setting specifies where files associated with a job will be temporarily stored while the job processes.

```
"JOBDATAFOLDER": "/var/cache/collab/jobdata"
```

SITE

If Anark Collaborate is deployed in multiple sites, specify the name - must be unique - to identify the site.

CONTENT SERVICE URL

Specify the **CONTENT_SERVICE_URL** to validate content in upload requests.

```
"CONTENT_SERVICE_URL": http://localhost:80
```

DOWNLOAD WORKER

To use the Download worker, a separate pod must be configured in PM2.json. This pod is named "downloadworker-0". The available environmental settings are detailed below.

PORT

By default, Download Worker will run on port 8092. To use a different port, change this option.

REDIS

This setting specifies the location where Redis is running.

```
"REDIS": "redis://<server>:6379"
```

REDIS USERNAME

If Redis is configured for authenticating with a username, then the **REDIS_USERNAME** property specifies the username used to connect to Redis.



REDIS PASSWORD

This setting is identical to the **REDISPASSWORD** setting in Anark Collaborate. Please refer to it above.

ENCRYPTION SECRET FILEPATH

This setting is identical to the **ENCRYPTION_SECRET_FILEPATH** setting in Anark Collaborate. Please refer to it above.

JOB DATA FOLDER

This setting specifies where files associated with a job will be temporarily stored while the job processes.

```
"JOBDATAFOLDER": "/var/cache/collab/jobdata"
```

SITE

If Anark Collaborate is deployed in multiple sites, specify the name - must be unique - to identify the site.

CONTENT SERVICE URL

Specify the **CONTENT_SERVICE_URL**, this will be used to get the content store information.

```
"CONTENT_SERVICE_URL": http://localhost:80
```

UPLOAD SERVICE

To use the Upload service, a separate Node server must be configured in PM2.json. This server is named "uploadservice-0". The available environmental settings are detailed below.

PORT

By default, Upload Service will run on port 2337. To use a different port, change this option.

SITE

If Anark Collaborate is deployed in multiple sites, specify the name - must be unique - to identify the site.

JOB DATA FOLDER

This setting specifies where files associated with a job will be temporarily stored while the job processes.

```
"JOBDATAFOLDER": "/var/cache/collab/jobdata"
```



REDIS

This setting specifies the location where Redis is running.

```
"REDIS": "redis://<server>:6379"
```

REDIS USERNAME

If Redis is configured for authenticating with a username, then the **REDIS_USERNAME** property specifies the username used to connect to Redis.

REDIS PASSWORD

This setting is identical to the **REDISPASSWORD** setting in Anark Collaborate. Please refer to it above.

ENCRYPTION SECRET FILEPATH

This setting is identical to the **ENCRYPTION_SECRET_FILEPATH** setting in Anark Collaborate. Please refer to it above.

AUTHENTICATION SERVICE URL

Specify the **AUTH_SERVICE_URL** to authenticate upload requests with the Anark Collaborate node server.

```
"AUTH_SERVICE_URL": "http://localhost:80"
```

CONTENT SERVICE URL

Specify the **CONTENT_SERVICE_URL** to validate content in upload requests.

```
"CONTENT_SERVICE_URL": http://localhost:80
```

UPLOAD WORKER

To use the Upload worker, which is required as part of upload service, a separate Node server must be configured in PM2.json. This server is named "uploadworker-0". The available environmental settings are detailed below.

PORT

By default, Upload Worker will run on port 8093. To use a different port, change this option.

SITE



Anark Collaborate Deployment

If Anark Collaborate is deployed in multiple sites, specify the name - must be unique - to identify the site.

JOB DATA FOLDER

This setting specifies where files associated with a job will be temporarily stored while the job processes.

```
"JOBDATAFOLDER": "/var/cache/collab/jobdata"
```

REDIS

This setting specifies the location where Redis is running.

```
"REDIS": "redis://<server>:6379"
```

REDIS USERNAME

If Redis is configured for authenticating with a username, then the **REDIS_USERNAME** property specifies the username used to connect to Redis.

REDIS PASSWORD

This setting is identical to the **REDISPASSWORD** setting in Anark Collaborate. Please refer to it above.

ENCRYPTION SECRET FILEPATH

This setting is identical to the **ENCRYPTION_SECRET_FILEPATH** setting in Anark Collaborate. Please refer to it above.

ENABLE REPLICATION

To enable or disable content replication between multiple sites, specify true or false for this setting.

```
"REPLICATIONENABLED": true
```

CONTENT SERVICE URL

Specify the **CONTENT_SERVICE_URL**, this will be used to get the content store information.

```
"CONTENT_SERVICE_URL": http://localhost:80
```

REPLICATION SERVICE

To use the Replication service, a separate Node server must be configured in PM2.json. This server is named "replicationservice-0". The available environmental settings are detailed below.

PORT



By default, Replication Service will run on port 3337. To use a different port, change this option.

SITE

If Anark Collaborate is deployed in multiple sites, specify the name - must be unique - to identify the site.

JOB DATA FOLDER

This setting specifies where files associated with a job will be temporarily stored while the job processes.

```
"JOBDATAFOLDER": "/var/cache/collab/jobdata"
```

REDIS

This setting specifies the location where Redis is running.

```
"REDIS": "redis://<server>:6379"
```

REDIS USERNAME

If Redis is configured for authenticating with a username, then the **REDIS_USERNAME** property specifies the username used to connect to Redis.

REDIS PASSWORD

This setting is identical to the **REDISPASSWORD** setting in Anark Collaborate. Please refer to it above.

ENCRYPTION SECRET FILEPATH

This setting is identical to the **ENCRYPTION_SECRET_FILEPATH** setting in Anark Collaborate. Please refer to it above.

CONTENT SERVICE URL

Specify the **CONTENT_SERVICE_URL** to validate content in upload requests with the Anark Collaborate node server.

```
"CONTENT_SERVICE_URL": http://localhost:80
```

AUTHENTICATION SERVICE URL

Specify the **CONTENT_SERVICE_URL**, this will be used to get the content store information.

```
"AUTH_SERVICE_URL": "http://localhost:80"
```

FILE SYSTEM REPLICATION



To enable content replication to different servers, use **FSSYNCCURLS** to specify the URLs of replica servers. The password can be encrypted. **FSSYNCTIMEOUT** is the amount of time, in milliseconds, before web service times out. Note that a user with "Content Author" role will be needed and specified in this field. If enabled, they should take the following form.

```
"FSSYNCCURLS": "https://<collabuser>:< password>@<host>:<port>;  
https://<collabuser>:<password>@<host>:<port>",  
"FSSYNCTIMEOUT": "<milliseconds>"
```

REPLICATION RETRY

Use **MAXRETRIES** to specify the maximum number of attempts the service will try to replicate content if replication fails.

Use **RETRYDELAY** to specify the time (in milliseconds) between each retry attempt. This number is doubled after each attempt. For example, using the specified 30 second delay and max retries of 3, will result in a retry attempt after 30 second, then after 60 seconds, and a final retry after 120 seconds.

```
"MAXRETRIES": 3  
"RETRYDELAY": 30000
```

MARKING SERVICE

To use Data Markings with content downloads, the Marking service is required. There are two services for data markings, one for image and one for PDF. Each data type requires a separate Node server and must be configured separately in PM2.json. These servers can be named "markingservice-image" and "markingservice-pdf", respectively. The available environmental settings are the same for both, unless otherwise specified, and are detailed below.

SERVICE TYPE

This setting specifies which file type Marking Service will be processing. The value should align with the service installation directory.

```
"name": "markingservice-image",  
"script": "/usr/local/collab/services/markingservice/src/index.js",  
"env": {  
  "SERVICE_TYPE": "image"  
}
```

or

```
"name": "markingservice-pdf",  
"script": "/usr/local/collab/services/markingservice/src/index.js",  
"env": {  
  "SERVICE_TYPE": "pdf"
```



```
}
```

PORT

By default, Marking Service will run on ports 8097 and 8098. To use a different port, change this option.

SITE

If Anark Collaborate is deployed in multiple sites, specify the name - must be unique - to identify the site.

JOB DATA FOLDER

This setting specifies where files associated with a job will be temporarily stored while the job processes.

```
"JOBDATAFOLDER": "/var/cache/collab/jobdata"
```

REDIS

This setting specifies the location where Redis is running.

```
"REDIS": "redis://<server>:6379"
```

REDIS USERNAME

If Redis is configured for authenticating with a username, then the **REDIS_USERNAME** property specifies the username used to connect to Redis.

REDIS PASSWORD

This setting is identical to the **REDISPASSWORD** setting in Anark Collaborate. Please refer to it above.

ENCRYPTION SECRET FILEPATH

This setting is identical to the **ENCRYPTION_SECRET_FILEPATH** setting in Anark Collaborate. Please refer to it above.

CONTENT SERVICE URL

Specify the **CONTENT_SERVICE_URL**, this will be used to get the content store information.

```
"CONTENT_SERVICE_URL": http://localhost:80
```

MARKING SERVICE TIMEOUT

Specify the **MARKING_SERVICE_TIMEOUT**, to set the time that marking service will process the file to apply markings. By default, it is set to 30 seconds and if it takes more than 30 seconds to apply the markings, download will fail. Use this setting to modify the timeout.



```
"MARKING_SERVICE_TIMEOUT": 60
```

CLOUD DEPLOYMENT

This setting allows the Marking service to configure itself for cloud deployment or virtual machine deployment. Set to false in your PM2.json.

```
"CLOUDDEPLOYMENT": false
```

LD LIBRARY PATH

For **markingservice-pdf** only, specify the installation location of the DotNetCORE binaries.

```
"LD_LIBRARY_PATH": "<install_dir>/services/markingservice/Pdf/Lib/APDFL18.0.3/DotNETCore/Binaries"
```

CONTENT SERVICE URL

Specify the **CONTENT_SERVICE_URL** to validate content in upload requests.

```
"CONTENT_SERVICE_URL": http://localhost:80
```

SANDBOX WORKER

To use Data Markings with content downloads, the Sandbox worker is required. A separate Node server must be configured in PM2.json. This server is named "sandboxworker-0". The available environmental settings are detailed below.

PORT

By default, Sandbox Worker will run on port 8099. To use a different port, change this option.

SITE

If Anark Collaborate is deployed in multiple sites, specify the name - must be unique - to identify the site.

REDIS

This setting specifies the location where Redis is running.

```
"REDIS": "redis://<server>:6379"
```

REDIS USERNAME



Anark Collaborate Deployment

If Redis is configured for authenticating with a username, then the **REDIS_USERNAME** property specifies the username used to connect to Redis.

REDIS PASSWORD

This setting is identical to the **REDISPASSWORD** setting in Anark Collaborate. Please refer to it above.

ENCRYPTION SECRET FILEPATH

This setting is identical to the **ENCRYPTION_SECRET_FILEPATH** setting in Anark Collaborate. Please refer to it above.



FILE SHARE WORKER

To use the File Share Worker, a separate pod must be configured in PM2.json. This pod is named "fileshareworker-0". The available environmental settings are detailed below.

PORT

By default, File Share Worker will run on port 4342. To use a different port, change this option.

REDIS

This setting specifies the location where Redis is running.

```
"REDIS": "redis://<server>:6379"
```

REDIS USERNAME

If Redis is configured for authenticating with a username, then the **REDIS_USERNAME** property specifies the username used to connect to Redis.

REDIS PASSWORD

This setting is identical to the **REDISPASSWORD** setting in Anark Collaborate. Please refer to it above.

ENCRYPTION SECRET FILEPATH

This setting is identical to the **ENCRYPTION_SECRET_FILEPATH** setting in Anark Collaborate. Please refer to it above.

SITE

If Anark Collaborate is deployed in multiple sites, specify the name - must be unique - to identify the site.

CONTENT SERVICE URL

Specify the **CONTENT_SERVICE_URL**, this will be used to get the content store information.

```
"CONTENT_SERVICE_URL": http://localhost:80
```

ENABLE REPLICATION

To enable or disable content replication between multiple sites, specify true or false for this setting.

```
"REPLICATIONENABLED": true
```



PUBLISH WORKER DATA

To use the Publish Worker Data, a separate pod must be configured in PM2.json. This pod is named “publishworkerdata-0”. The available environmental settings are detailed below.

PORT

By default, Publish Worker Data will run on port 4341. To use a different port, change this option.

REDIS

This setting specifies the location where Redis is running.

```
“REDIS”: “redis://<server>:6379”
```

REDIS USERNAME

If Redis is configured for authenticating with a username, then the **REDIS_USERNAME** property specifies the username used to connect to Redis.

REDIS PASSWORD

This setting is identical to the **REDISPASSWORD** setting in Anark Collaborate. Please refer to it above.

ENCRYPTION SECRET FILEPATH

This setting is identical to the **ENCRYPTION_SECRET_FILEPATH** setting in Anark Collaborate. Please refer to it above.

SITE

If Anark Collaborate is deployed in multiple sites, specify the name - must be unique - to identify the site.

JOB DATA FOLDER

This setting specifies where files associated with a job will be temporarily stored while the job processes.

```
“JOBDATAFOLDER”: “/var/cache/collob/jobdata”
```

CONTENT SERVICE URL

Specify the **CONTENT_SERVICE_URL** to validate content in upload requests.

```
“CONTENT_SERVICE_URL”: http://localhost:80
```



PUBLISH CONTROLLER

To use the publishing service, Publish Controller needs to be deployed and configured in addition to the Publish worker [deployment](#) described above. A separate Node server must be configured in PM2.json for the Publish Controller. This server is named “publishcontroller-0”. The available environmental settings are detailed below.

PORT

By default, Publish Controller will run on port 4340. To use a different port, change this option.

REDIS

This setting specifies the location where Redis is running.

```
“REDIS”: “redis://<server>:6379”
```

REDIS USERNAME

If Redis is configured for authenticating with a username, then the **REDIS_USERNAME** property specifies the username used to connect to Redis.

REDIS PASSWORD

This setting is identical to the **REDISPASSWORD** setting in Anark Collaborate. Please refer to it above.

ENCRYPTION SECRET FILEPATH

This setting is identical to the **ENCRYPTION_SECRET_FILEPATH** setting in Anark Collaborate. Please refer to it above.

JOB DATA FOLDER

This setting specifies where files associated with a job will be temporarily stored while the job processes.

```
“JOBDATAFOLDER”: “/var/cache/anark/jobdata”
```

SITE

If Anark Collaborate is deployed in multiple sites, specify the name - must be unique - to identify the site.

CONTENT SERVICE URL

Specify the **CONTENT_SERVICE_URL**, this will be used to get the content store information.

```
“CONTENT_SERVICE_URL”: http://localhost:80
```



CORE SERVICE URL

Specify the **CORE_SERVICE_URL**, this will be used to get the tenant preferences.

```
"CORE_SERVICE_URL": http://localhost:80
```

MESSAGE WORKER

The Message Worker will alert the cluster of any changes to tenant preferences. A separate Node server must be configured in PM2.json. This server is named "messageworker-0". The available environmental settings are detailed below.

PORT

By default, Message Worker will run on port 9000. To use a different port, change this option.

SITE

If Anark Collaborate is deployed in multiple sites, specify the name - must be unique - to identify the site.

DATABASE

The **DB** property specifies the MongoDB connection string. In general, it should take the following form. The High Availability section below has details on how to configure MongoDB replica sets.

```
mongodb://<host>:<port>/<database >
```

DATABASE USER

The **DBUSER** property specifies the username to use when connecting to MongoDB.

DATABASE PASSWORD

The **DBPASSWORD** property specifies the encrypted password associated with **DBUSER** which is required to authenticate with MongoDB. When specifying **DBPASSWORD**, the secret used to encrypt the password must be provided via the **ENCRYPTION_SECRET_FILEPATH** property.

REDIS

This setting specifies the location where Redis is running.

```
"REDIS": "redis://<server>:6379"
```

REDIS USERNAME



Anark Collaborate Deployment

If Redis is configured for authenticating with a username, then the **REDIS_USERNAME** property specifies the username used to connect to Redis.

REDIS PASSWORD

This setting is identical to the **REDISPASSWORD** setting in Anark Collaborate. Please refer to it above.

ENCRYPTION SECRET FILEPATH

This setting is identical to the **ENCRYPTION_SECRET_FILEPATH** setting in Anark Collaborate. Please refer to it above.

CONTENT SERVICE URL

Specify the **CONTENT_SERVICE_URL** to validate content in upload requests.

```
"CONTENT_SERVICE_URL": http://localhost:80
```

ADMIN PORTAL – SYSTEM PREFERENCES

The Admin portal *System* pane enables users with the **MANAGE-PREFERENCES** permission to modify most of the system configurations without restarting the system. Preferences are split into seven categories, *Content*, *Users*, *Work Management*, *Publishing*, *Notifications*, *Search*, and *Miscellaneous*.

CONTENT

System Preferences *Content* includes all the necessary options to configure how contents are displayed and stored in Anark Collaborate plus options to configure access control for content.

CONTENT PROPERTIES SCHEMA

To enforce publishing of content with certain metadata, the **Content Properties Schema** can be configured to require all published content to have specific properties with specific value types. Follow the below syntax to define as many requirements as needed.

- Property Name - A name to give the property without spaces. Ex. publishDate
- Property Type:
 - String & optional enumerations
 - Boolean
 - Date
 - Integer
 - Decimal
- Required - Can be "true" or "false"



Anark Collaborate Deployment

- Display Name - Represents how the published metadata will be shown next to the Content in Anark Collaborate.

If a property type is string, an additional field is available, but optional, for specifying enumerations, or a list of acceptable values. For example:

- Property Name: releaseStatus
- Property Type: string
- Required: true
- Display Name: Release Status
- Options: released, prerelease, beta, alpha

Content Properties Schema

Decimal : decimal | false x Boolean : boolean | false x Date : date | false x
Integer : integer | false x Enum : string | false x

Property Name

Property Type

Required

Display Name

Options

[Add Property](#)

A property can be edited by clicking on the property directly. For example, clicking on one of the properties in the blue box below will change the form to allow for updating that property. Property edits must be saved by clicking the “Update Property” button.

Content Properties Schema

Decimal : decimal | false x Boolean : boolean | false x Date : date | false x
Integer : integer | false x Enum : string | false x

Property Name

Property Type

Required

Display Name

Options

[Discard Changes](#) [Update Property](#)

SHOWN CONTENT METADATA



The preference **Content Properties To Show** represents the content metadata that can be displayed for each published content in Anark Collaborate. This can be configured for all users or users with certain roles. Use the role name text box to enter the role to specify the preference for a certain role. Specify whether the property key needs to be shown along with the value using the middle dropdown. Select the content metadata that needs to be shown using the “Select Properties To Show...” dropdown. It is recommended to include any metadata that will help in identifying content, especially if multiple versions have been published. For example:

Content Properties To Show

[Show Property](#)

Example	Description	Example
false PUBLISH DATE	All users will see the PUBLISH DATE but will not see the key descriptor.	Content A 20191102
collaborator : true DESCRIPTION	Collaborator will see DESCRIPTION with a key descriptor plus what all users have configured above.	Content A Description: Content a is a... 20191102
supplier : true REVISION, PUBLISH DATE	Supplier will see REVISION, PUBLISH DATE which overrides what all users see.	Content A Revision: A Publish Date: 20191102

INHERIT CONTENT ACCESS FROM ITEM

In some cases, it is desired to have different content access privileges when that content is used by an Activity or Work Instruction. Enabling this preference gives content access to users participating in an Activity or Work Instruction. If the user has access to viewing a Work Instruction, enabling this preference will allow them to view the content items included in that Work Instruction even if they do not have access to those content items. The same is true for an Activity, that is, if a user has access to the Activity they inherit access to the content item included in that Activity.



ACCESS CONTROL WEB SERVICE

If Anark Collaborate is integrated to use external access control for content items, the **Access Control Web Service URL** and **Timeout** preferences are used to specify the web service for querying content access checks and when the check times out, in milliseconds.

ACTIVITIES

SHOWN ACTIVITY METADATA

The preference **Activity Properties To Show** represents the activity metadata that can be displayed for each activity in search results. Activity metadata can be configured to only be shown for specific roles. The rule syntax mimics that of content metadata described above under **Shown Content Metadata**.

Only certain fields can be configured: description, start date, and end date. Additionally, declared properties can be edited by clicking on the property (in grey) and then the “Update Property” button.

USERS

The following preferences can be configured for Anark Collaborate Users.

SHOWN USER METADATA

The preference **User Properties To Show** represents the user metadata that can be displayed for each user on the activity page, in search results, and on the group page - both Anark Collaborate users and External users. User metadata can be configured to only be shown for specific roles. The rule syntax mimics that of content metadata described above under **Shown Content Metadata**.

Only certain fields can be configured: userid, email, organization, department, title, and active. Additionally, declared properties can be edited by clicking on the property (in grey) and then the “Update Property” button.

USER SITES

Many customers configure Anark Collaborate to run in multiple sites, e.g. North America, Europe, and Asia. This preference allows Admins to define all the available sites. Users will then have an option to select a site for which they are associated (during User creation or editing). To define a site simply type the site name and press “Tab” or “Enter”. Multiple sites will appear like the below example:



GUEST USERS

Anark Collaborate can be configured to allow for guest user access if it is also configured to use external authentication. This preference defines how guest user headers will map to the user data model when stored in



the user session. The Custom property allows for storing non-mapped headers on the session. Both **Name** and **User Id** are required for guest user access.

Additionally, select one or more system or custom roles that guest users will have, when using Anark Collaborate.

Guest User Configuration

Defined header mappings

Source header name

User Model Property

Add Header Mapping

Guest's Roles

Guest's Roles

WORK INSTRUCTIONS

There are several preferences for configuring Work Management preferences in Anark Collaborate. These preferences do **not** need to be modified if the organization is not using the Work Management capability.

WORK INSTRUCTION PROPERTIES SCHEMA

To enforce creation of Work Instructions with certain metadata, the **Work Instruction Properties Schema** can be configured to require all newly created Work Instructions to have specific properties with specific value types. The syntax is identical to **Content Properties Schema**. Additionally, declared properties can be edited by clicking on the property (in grey) and then the “Update Property” button.

SHOWN WORK INSTRUCTION METADATA

The preference **Work Instruction Properties to Show** represents the metadata that can be displayed for each Work Instruction in Anark Collaborate. This parameter can be configured to show certain metadata based on user roles and has identical syntax to **Shown Content Metadata**. Additionally, declared properties can be edited by clicking on the property (in grey) and then the “Update Property” button.

WORK INSTRUCTION EXECUTION PROPERTIES SCHEMA

To enforce execution of Work Instructions with certain metadata, the **Work Instruction Execution Properties Schema** can be configured to require users to enter Work Instruction Execution information before executing. The syntax is identical to **Content Properties Schema**. Additionally, declared properties can be edited by clicking on the property (in grey) and then the “Update Property” button.

WORK INSTRUCTION PLAYER LABELS

The Work Instruction execution and authoring interface separates important information into specific panes, each with their own label. These labels, as shown in the figure below, can be configured under **Work Instruction Player Labels**.



Anark Collaborate Deployment

Work Item Player Labels

Steps	Process Steps
Description	Description
Safety	Safety & Compliance
Contexts	Content
Verifications	Verifications

1
1.1
1.1.1
1.2
2
3

PROCESS STEPS

SAMPLE WORK ITEM 1

DESCRIPTION

Verify you have all required materials

SAFETY & COMPLIANCE

Make sure you are wearing your **helmet!!!**

VERIFICATIONS

TEXT 1 Show Details
Text

DROPDOWN 1 Show Details
Dropdown

WORK INSTRUCTION ACCESS CONTROL WEB SERVICE

If Anark Collaborate is integrated to use an external access control framework for work instructions, the **Work Instruction Access Control Web Service URL** and **Timeout** preferences are used to specify the web service for querying Work Instruction access checks and when the check times out, in milliseconds.

PUBLISHING

The following preferences are used when uploading Content to Anark Collaborate (currently available as “Uploads” when authoring a Work Instruction), or when downloading Content with watermarks.

UPLOAD SETTINGS

Upload Settings are preferences for establishing configuration options for viewing uploaded files.

Illustration Height: The height of published illustration images for model views

Illustration Width: The width of published illustration images for model views

Image Background Color: The background color of published images for model views

Image Height: The height of published images for model views

Image Width: The width of published images for model views

Image Thumbnail Background Color: The background color of the content thumbnail



Image Thumbnail Height: The height of the content thumbnail

Image Thumbnail Width: The width of the content thumbnail

WATERMARKS

Anark Collaborate supports downloading Content to your local device after applying a watermark (if applicable). The system supports multiple watermarks and each can be configured under **Watermark Preferences**. Additionally, watermarks can be edited by clicking on the watermark (in grey) and then the “Update Watermark” button.

Watermarks are fully customizable and can be configured with the following preferences:

Preference	Description	Example & Default
Text	The text to display in the watermark. Text can include any of the three dynamic variables: Requesting User’s Username: <code>\${uname}</code> Requesting User’s IP Address: <code>\${ip}</code> Requested at Timestamp: <code>\${ts}</code>	User: <code>\${uname}</code> IP: <code>\${ip}</code> Timestamp: <code>\${ts}</code>
Font Opacity	0 to 1 number representing the visibility of the watermark. 0 is invisible.	0.7
Font	The font style/family to render the text with.	Lucida Console
Font Color	The color of rendered text, in Hex notation.	#000000
Font Size	The size of rendered text.	20
Text Align	The alignment of rendered text. Three available options: ‘left’, ‘center’, or ‘right’	center
Rotation	The rotation of rendered text, from 0 to 360 degrees.	0



Horizontal Position	Relative position for rendering text along the horizontal axis. Relative renditions require one of three options: 'left', 'center', or 'right'.	center
Vertical Position	Relative position for rendering text along the vertical axis. Relative renditions require one of three options: 'top', 'center', or 'bottom'.	center
Horizontal Offset	The offset, in pixels, from the chosen relative position.	0
Vertical Offset	The offset, in pixels, from the chosen relative position.	0
Start Page	Only applicable if downloading a PDF. The first page to begin applying the watermark.	0
End Page	Only applicable if downloading a PDF. The last page to apply the watermark. If specified number is greater than the total number of pages, the last possible page will be used.	1000



Anark Collaborate Deployment

Watermark Preferences

User: \${uname} IP: \${ip} Times... x

Text: Write your text here...

Font Opacity: 1

Font: Lucida Console

Font Color: #000000

Font Size: 20

Text Align: Center

Rotation: 0

Horizontal Position: Center

Vertical Position: Center

Horizontal Offset: 0

Vertical Offset: 0

Start Page [PDF]: Enter a page number if watermark will be applied to a PDF
1

End Page [PDF]: Enter a page number if watermark will be applied to a PDF
1000

Add Watermark

DATA MARKINGS

Anark Collaborate supports adding data markings to PDF or image content item components when downloading. During the download process, a margin is added around the PDF or image where markings are applied. Markings cannot be removed. Where markings are applied, is determined by the markings template defined in the Anark Collaborate System Preferences.

A markings template may consist of several text boxes positioned in the page margins. Text boxes support rich text including font, bold, italic, underline, and color; however, defining the rendered text can be done either by entering the string of text directly or by writing a JavaScript expression that returns a string of text. Each page margin – top, bottom, left, or right – can have as many text boxes as desired, each with its own rich text.

Text boxes also support dynamic variables in the string of text. Using the syntax `${<variable>}`, strings can be evaluated to dynamically add important information about the content, the user accessing the content, the timestamp when the content download was requested, and the internet protocol address from which the user is requesting the download. Details pertaining to the exact information available through dynamic variables, is mentioned later in this document.

For example:

To apply a rich text data marking without JavaScript, specify the following rich text:

```
Content was downloaded by ${user.userId} at ${ts} and from ${ip}
```



Result:

Content was downloaded by **nathan** at **Wed May 11 2022 20:28:44 GMT+0000 (Coordinated Universal Time)** and from [172.20.0.1](#)

To apply a rich text data marking with JavaScript, write a conditional expression like:

```
if (user.organization.includes('Anark') {  
  return [  
    {text:'Hello\n', color: 'blue', bold: true},  
    {text: 'World.', italic: true, font: 'Courier New'},  
  ]  
}
```

Result:

Hello
World.

DEFINING DATA MARKING TEMPLATES

Data marking templates are defined per component type. One template will be used for all components of that type. Only PDFs and images are supported at this time. In the Anark Collaborate System Preferences, marking templates are defined in specified page margins, with one or more text boxes. Using the following form fields, a template can be defined.

The form is divided into two sections. The top section defines the size of each margin relative to the document being downloaded. 50% top margin would be half the total width of the component document or image. It also gives a name to the template and binds a component type to the template.



Anark Collaborate Deployment

The screenshot shows a configuration interface for a PDF template. At the top, there is a 'Markings Template' field containing 'pdf: PDF' and 'image: Image'. Below this is a 'Name' field with 'PDF' and a 'Component Type' dropdown set to 'PDF'. A tabbed interface allows selecting a margin: 'Top Margin' (selected), 'Right Margin', 'Bottom Margin', and 'Left Margin'. The 'Top Margin [%]' is set to 15. Below the margin tabs is a 'Text Boxes' section with a list containing 'Content Name' and 'Content Description'. The 'Content Name' box is selected, showing its configuration: 'Name' is 'Content Name', 'Type' is 'Text', 'Minimum Lines in Box' is 1, and 'Rich Text' contains 'Arial' with formatting icons and dynamic variables like '\$(content.name)', 'Revision: \$(content.properties.revision)', and 'Business: \$(content.properties.site)'. 'Text Alignment' is set to 'Left', 'Width [%]' is 50, and 'Height [%]' is 100. At the bottom right are buttons for 'Discard Changes', 'Update Template', and 'Update Text Box'.

The bottom half allows for text box definitions. Each text box has the following attributes:

Name – an identifier for the template. Text box names are not included in the applied data markings.

Type – can either be “Text” or “JavaScript”. When text, the preceding field is “Rich Text”. When JavaScript, the preceding field changes to “JavaScript”.

Minimum Lines in Box – the minimum number of lines of text that data markings should try to fit to. Font size will scale as needed to meet the number of lines specified.

Rich Text – a rich text editor that allows for static text and dynamic variables with rich formatting – font, bold, italic, underline, and font color. Note: when applying rich formatting to dynamic variables, the formatting must also be applied to the wrapping syntax identifier “\$()” as seen in the picture above. If not correctly applied, the dynamic variable will still be evaluated but rich formatting will not be applied.

JavaScript – an editor for entering JavaScript. JavaScript will be executed server-side at runtime when markings are applied. When executing, the JavaScript will only have access to the available Dynamic Variables listed below. See below section for more information.

Text Alignment – align the text Left, Center, or Right.

Width & Height – the relative percentage of available width or height within the chosen margin with respect to other text boxes. For example, two text boxes in the bottom margin may each have 100% height and 50% width. Four text boxes in the left margin may each have 100% width and 25% height such that the combined widths and heights does not exceed 100%.



DYNAMIC VARIABLES IN DATA MARKINGS

Dynamic Variables add user and content specific information to a data marking. The following information is available through dynamic variables:

content.name – the name of the content item which contains the components that data markings are applied to.

content.description – the description of the content item which contains the components that data markings are applied to.

content.properties – an object of key-value pairs - { "propName": "propValue" } - for each custom content property specified when publishing the content item. Ex, { "revision": "A", "version": "1.3" }

user.userId – the login id (userId) of the user accessing the content item.

user.name – the name of the user.

user.organization – the organization (string) or organizations (array) for which the user belongs to.

user.roles – the role names associated with the user.

user.site – the site the user is registered to.

user.sessionInfo – an object of key-value pairs containing custom session headers. Ex, { "plmId": "12345" }

ts – the timestamp at which a user requests a download of components with data markings.

ip – the internet protocol address from where the user requested the download of components with data markings.

JAVASCRIPT IN DATA MARKING TEMPLATES

Data marking templates have the flexibility to display different information based on the user accessing the content item and/or the content item being accessed. For example, if sharing a content with a supplier, you may want different markings for users within your organization than users within the supplier's organization. Since these conditionals may vary between users, locations, content items, or more, JavaScript is not executed until runtime when the markings are being applied.

For enhanced security, data markings with JavaScript are executed in a sandbox and asynchronous JavaScript will not be executed. This requires the JavaScript to return information in specific formats to be used in the markings. Users defining marking templates can either return a single string of static text or dynamic variables, or an array of objects containing strings with rich formatting.

Example 1:

```
if (user.organization.includes('Anark')) {  
  return [  
    {text:'Hello\n', color: 'blue', bold: true},  
    {text:'World.', italic: true, font: 'Courier New'},  
    {text:'My name is ${user.userId}'},  
  ]  
}
```



```
}

```

Example 2:

```
if (user.organization.includes('Anark')) {
    return 'Hello world.'
}
```

NOTIFICATIONS

Anark Collaborate is configurable to send system email notifications to individuals when certain changes have been made. The system can be configured to use an external SMTP server.

ENABLE EMAIL NOTIFICATIONS

This preference specifies what system actions will send a notification email.

Value	Action Description	Additional Details
CREATE ACTIVITY	This value enables sending notifications when creating an activity.	All users, group members, and activity owners participating in the activity will receive a notification.
EDIT ACTIVITY	This value enables sending notifications when an activity has been modified.	
DELETE ACTIVITY	This value enables sending notifications when an activity has been deleted.	
CREATE CONVERSATION	This value enables sending notifications when a conversation has been created.	All users, group members, and conversation owner participating in the conversation and activity owners will receive a notification.
EDIT CONVERSATION	This value enables sending notifications when a conversation has been modified.	



DELETE CONVERSATION	This value enables sending notifications when a conversation has been deleted.	
CREATE COMMENT	This value enables sending notifications when a comment has been added to a conversation.	All users, group members, and conversation owner participating in the conversation and activity owners will receive a notification.
EDIT COMMENT	This value enables sending notifications when a comment has been modified.	
DELETE COMMENT	This value enables sending notifications when a comment has been deleted.	

SMTP SERVER

This preference specifies the connection URL to connect to external SMTP server for sending email notifications. Leave this value as an empty string "" if using Sendmail. SMTP connection URL should be specified in the below format.

```
"smtp(s)://user:password@server:port/?pool=true"
```

User and password parameter can be omitted if not needed. To debug connection issues, query parameter "logger" and "debug" can be used. Setting "debug" query parameter to "true" logs the SMTP traffic in Anark Collaborate logfile. "logger" query parameter logs other useful debugging messages in the Anark Collaborate log file.

```
"smtp(s)://user:password@server:port/?pool=true&logger=true&debug=true"
```

SEND FROM EMAIL ADDRESS

When notification emails are enabled, this property specifies the email address notifications will be sent from. First provide the senders name followed by the email address. In the following example "Anark Collaborate Notifications" is the senders name, and "no-reply-collab@company.com" is the senders email address.

```
"Anark Collaborate Notifications no-reply-collab@company.com"
```

NOTIFICATION EMAIL URLS

If Anark Collaborate is deployed in multiple sites and each site's user uses a different URL to access Anark Collaborate, then this preference can be configured to specify the base URL for each site so that the correct URL based on user's site is shown in the notification emails. For example, an Anark Collaborate Activity can include



users from multiple sites. Consider a user who belongs to the Europe site, is subscribed to an Activity, and gets emails when a Conversation is created. With this preference, an email notification for a newly created Conversation by a user who belongs to the Asia site would include a link to that Conversation but accessed through the European site's Anark Collaborate URL. If this preference was not set, the notification email will show the Asia site's Anark Collaborate URL.

SEARCH

SEARCHABLE AND SORTABLE PROPERTIES

Search within Anark Collaborate is made possible using MongoDB's indexing and querying mechanics. For each of the searchable items (Content, Activities, Users, Work Instructions), admins can define fields which are *searchable* and fields which are *sortable*. This information is used to create the indexes and the data structures necessary to support fast and efficient searches.

The reason *all* fields are not searchable and sortable by default is because this would require the creation of potentially hundreds of indexes. This would create unnecessary strain on the memory and disk usage for the database server, causing *all* database queries to become very slow. MongoDB's indexing works best when it is fine tuned to the specific types of queries you wish to perform. It is strongly advised that you do not make all fields searchable or sortable on a just-in-case basis as there are serious performance implications for doing this.

Note: The general rule about indexing is the more indexes, the more RAM and disk space required, and the slower things become all around - not just searching.

Each item has a set of predefined fields which the user can search on, and if the item supports custom properties, any string custom property can be defined as searchable. System Preference UI allows the admin to specify different or same searchable field with different sort combinations, all the individual words contained within those fields are aggregated into a special "keywords" column. Any time a user performs a search, we query this keywords column to find any matches.

The same properties can be defined as both searchable and sortable properties. Sortable properties determine the order in which search results are returned. Admins can define as many sort combinations as they wish - each sort combination can contain up to 2 fields (ex. sort by ColumnA then by ColumnB). The default sort order is by *_id* descending - which roughly equates to "newest first". There is no need to create an index to support the default sort - the system creates this automatically.

Example: A user or department might need to sort content items first by "documentid" and then by "version" so that any documents with the same documentid are further sorted by version. An admin would create a sort combination that contains documentid first, followed by version (the order matters). If another user or department wishes to sort by "version" first and then "documentid", a separate sort combination will need to be created. If someone wishes to sort by just "version" or "documentid", two separate sort indexes will need to be created to support both of those.

For every sort combination, updateSearchKeywords.js script will generate the proper indexes necessary to support fast and efficient queries. However, it is highly recommended that you create as few sort settings as necessary to meet the needs of your users. Creating unnecessary sort combinations will create strain on the database and can have a detrimental impact on the overall database performance.



Anark Collaborate Deployment

Whenever an admin makes changes to any of the sort settings, those changes are NOT immediately applied to the database. Instead, an administrator must manually run a script (`updateSearchKeywords.js`) to synchronize the settings with the database. The reason for this manual step is because the process of updating keywords and indexes will likely consume a lot of server resources. Admins can control when this work is done to impact as few users as possible (e.g., nights and weekends).

The script executes two separate tasks:

1. Update all keywords to ensure all *searchable* fields are being indexed properly. This involves updating every document in the database.
2. Create and delete indexes based on the *sortable* field combinations. Only new indexes are created, and old indexes removed. Modified indexes are dropped and recreated, and unchanged indexes remain untouched.

ACTIVITY

Anark Collaborate supports searching for Activities by metadata defined in the **Searchable Activity Properties** field. By default, the system is configured to search *title*. The app can also be configured to search *description*. Any field can be specified as a custom sort order.

Changes made to Searchable Activity Properties will only apply to newly created activities. To apply changes to existing activities, follow the instructions at the top of the file `/usr/local/collab/scripts/updateSearchKeywords.js`. **WARNING:** Running this script can take a while depending on the number of activities. We recommend running this script when the system is under minimal load.

Example:

*Searchable Activity Properties	<input type="text" value="searchable props"/>
Select or Type Properties to Search*	<input type="text" value="description title"/>
Select or Type Properties to Sort	<input type="text" value="Select..."/>
Index Name	<input type="text" value="searchable props"/>
	<input type="button" value="Discard Changes"/> <input type="button" value="Update Index"/>



CONTENT ITEM

Anark Collaborate supports searching for Content by metadata defined in the **Searchable Content Properties** field. By default, the system is configured to search for *name* and *description*. The app can also be configured to search and sort any other properties.

Changes made to Searchable Content Properties will only apply to newly published content. To apply changes to existing content, follow the instructions at the top of the file `/usr/local/collab/scripts/updateSearchKeywords.js`. **WARNING:** Running this script can take a while depending on the number of published content items. We recommend running this script when the system is under minimal load.

WORK INSTRUCTION

Anark Collaborate supports searching for Work Instructions by metadata defined in the **Searchable Work Instruction Properties** field. By default, the system is configured to search for *title* and *description*. The app can also be configured to search and sort any other properties.

Please note that changes made to Searchable Work Instruction Properties will only apply to newly created work instructions. To apply changes to existing work instructions, follow the instructions at the top of the file `/usr/local/collab/scripts/updateSearchKeywords.js`. **WARNING:** Running this script can take a while depending on the number of work instructions. We recommend running this script when the system is under minimal load.

USER

Anark Collaborate supports searching for Users by user metadata defined in the **Searchable User Properties** field. By default, the system is configured to search for *name*. The app can also be configured to search and sort *email* and *user id*.

Please note that changes made to Searchable User Properties will only apply to newly created users. To apply changes to existing users, follow the instructions at the top of the file `/usr/local/collab/scripts/updateSearchKeywords.js`. **WARNING:** Running this script can take a while depending on the number of users. We recommend running this script when the system is under minimal load.

GROUP

Anark Collaborate supports searching for Groups by group metadata defined in the **Searchable Group Properties** field. By default, the system is configured to search for *name*. The app can also be configured to search and sort *description*.

Please note that changes made to Searchable Group Properties will only apply to newly created groups. To apply changes to existing groups, follow the instructions at the top of the file `/usr/local/collab/scripts/updateSearchKeywords.js`. **WARNING:** Running this script can take a while depending on the number of users. We recommend running this script when the system is under minimal load.



MINIMUM CHARACTERS FOR SEARCH

Anark Collaborate supports a search restriction that will disable searching for any item unless a minimum number of characters have been typed. If the preference is set to zero, then no restriction is applied. For example, if the preference is set to four, then all users will have to type at least four characters to perform the search. Anark Collaborate apps will disable the search button until the user enters the required number of minimum characters. Additionally, blank searches will not be possible.

USER SEARCH RESTRICTIONS

An Anark Collaborate user can search for all the users in the system by default. User search can be configured to restrict what other organizations can be searched to find users in addition to the organization they belong to. These user search restrictions can be configured using roles, and the Admin can specify the additional organizations that the user in a specific role can search for in addition to their organization. If a user belongs to multiple roles, and rules are specified for both the roles, then user can search for users within their organization and the selected organizations of both rules.

SEARCH ACCESS CONTROL WEB SERVICE

If Anark Collaborate is integrated to use external access control for searching content items and/or work instructions, the **Search Work Instruction Access URL**, **Search Content Access Web Access URL**, and respective **Timeout** preferences, are used to specify the web service for querying search result access checks and when the check times out, in milliseconds.

MISCELLANEOUS

The following preferences are the remaining preferences that do not belong in any of the previous categories.

ITEM ACCESS CACHING

If Content or Work Instruction access control in Anark Collaborate is managed by an external system, Anark Collaborate invokes an external web service to check the access whenever a user tries to view or collaborate with it. Access can be cached if needed for a specified period for performance reasons (to reduce the number of external web service calls). Anark Collaborate will always check the external system for access when user tries to view the item irrespective of this setting.

To enable access caching on the server simply change the property **Enable Item Access Cache** from false to true and add an expiration time using the property **Item Access Cache Timeout** (in seconds).

JOB TTL

This preference specifies the time that needs to elapse in seconds once the job is started before temporary files created for the job will be deleted.

LOGGING

Logging Level specifies the depth of logging to use. This field can be set to one of the following:



Anark Collaborate Deployment

- Error
- Warn
- Info
- Debug

Each setting will log every event for itself and all settings above. For example, a setting of Info will log any Info and any Error or Warning log messages. It will not log Debug messages.

DEFINE HELP FILES

Depending on who is using the system, help files can be specified for specific user roles using the same rule syntax as described in Content Metadata above. Help files should be stored in `../collab/apps/help/`. When defining a help file, specify a role name if applicable, and type the file name plus extension. Once finished, click “Update Help File”. Additionally, declared files can be edited by clicking on the file (in grey) and then the “Update Help File” button.

Define Help Files

User Reference.pdf x

Role

*File Name

Type a file name...

Add Help File

Example	Description
Supplier : Supplier.pdf	Users with SUPPLIER role will see <i>Supplier.pdf</i> and <i>User Reference.pdf</i>
Acmeadmin : Admin.pdf	Users with ACMEADMIN role will see <i>Admin.pdf</i> and <i>User References.pdf</i>
User Reference.pdf	All other users will only see <i>User Reference.pdf</i>

DEFINE CUSTOM HELP LINKS

Depending on who is using the system, help links can be specified for specific user roles using the same rule syntax as described in Content Metadata above. When defining a help link, specify a role name if applicable, and add the link address plus a display name. Once finished, click “Add Custom Link”. Additionally, declared links can be edited by clicking on the link (in grey) and then the “Update Custom Link” button.



Anark Collaborate Deployment

Define Custom
Links

viewer : Company Tech Suppo... x

Role Name

x | v

Display Name

Type a link display name...

Link Address

https://...

Add Custom Link

Clear Form



IDENTITY MANAGEMENT

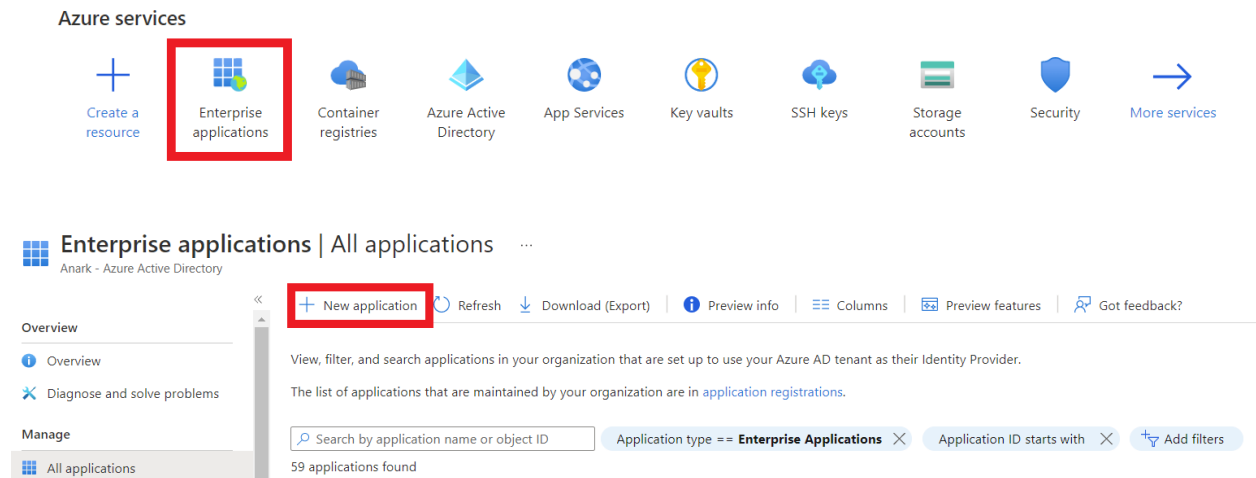
In addition to local authentication, Anark Collaborate also supports single sign-on using the SAML 2.0 protocol.

SAML stands for Security Assertion Markup Language and is an established protocol for handling identity verification with an Identity Provider. The Service Provider (Anark Collaborate) and the Identity Provider will communicate with each other through a series of cryptographically signed XML documents that are exchanged during the login process and contain all the relevant user information. More information about SAML can be found here: https://en.wikipedia.org/wiki/Security_Assertion_Markup_Language

GETTING STARTED

SAML for Anark Collaborate has been extensively tested with Azure Active Directory (AD), though it should work with any Identity Provider (IdP) that adheres to the SAML 2.0 protocol. This guide will demonstrate how to integrate Anark Collaborate with Azure AD, so users can login to Anark Collaborate using their Azure AD credentials.

The first thing to do is login to the Azure portal. Here, you will be able to create a new enterprise application.



Once your application has been created, navigate to the Single sign-on section, which can be found on the left-hand side of the application home screen.



Anark Collaborate Deployment

Overview | Deployment Plan | Manage | Properties | Owners | Roles and administrators | Users and groups | **Single sign-on** | Provisioning | Self-service | Custom security attributes

Upload metadata file | Change single sign-on mode | Test this application | Got feedback?

Set up Single Sign-On with SAML

An SSO implementation based on federation protocols improves security, reliability, and end user experiences and is easier to implement. Choose SAML single sign-on whenever possible for existing applications that do not use OpenID Connect or OAuth. [Learn more.](#)

Read the [configuration guide](#) for help integrating Anark Cloud SAML SSO Dev2.

1 Basic SAML Configuration Edit

Identifier (Entity ID)	https://	.com/
Reply URL (Assertion Consumer Service URL)	https://	.com/samlreturn
Sign on URL	https://	.com/
Relay State (Optional)	Optional	
Logout Url (Optional)	https://	/samllogout

In Anark Collaborate, you can find the “Identity Management” section when you select “System Preferences” from the “More” dropdown menu.

ANARK Search | Home | Create | More

ANARK CORE™ MBEWEB System | **Identity Management**

SAML

*SSO Login URL

*Service Provider Issuer Id (Entity ID)

System Preferences | Help | © 2022 Anark Corporation. All rights reserved.

Now we have everything in place to begin configuring Anark Collaborate to allow single sign-on with Azure AD using SAML.

BASIC SET UP

In the very first section of the application home screen in the Azure portal, titled “Basic SAML Configuration”, click “Edit” in the upper right corner.

Overview | Deployment Plan | Manage | Properties | Owners | Roles and administrators | Users and groups | **Single sign-on** | Provisioning | Self-service | Custom security attributes

Upload metadata file | Change single sign-on mode | Test this application | Got feedback?

Set up Single Sign-On with SAML

An SSO implementation based on federation protocols improves security, reliability, and end user experiences and is easier to implement. Choose SAML single sign-on whenever possible for existing applications that do not use OpenID Connect or OAuth. [Learn more.](#)

Read the [configuration guide](#) for help integrating Anark Cloud SAML SSO Dev2.

1 Basic SAML Configuration Edit

Identifier (Entity ID)	https://	.com/
Reply URL (Assertion Consumer Service URL)	https://	.com/samlreturn
Sign on URL	https://	.com/
Relay State (Optional)	Optional	
Logout Url (Optional)	https://	/samllogout



Anark Collaborate Deployment

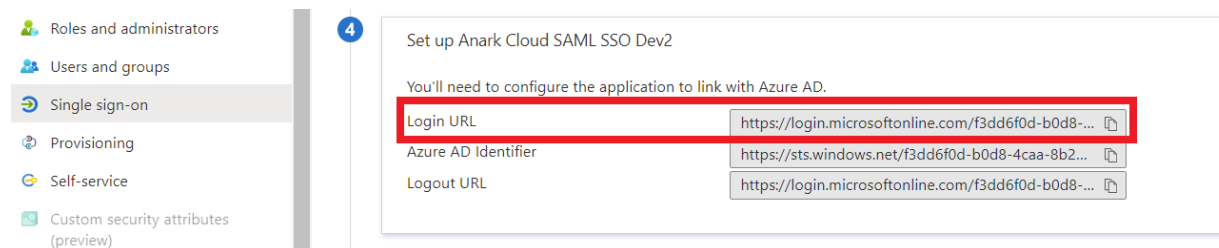
In this section, we are only going to focus on the **Entity ID**, the **Reply URL**. The **Sign on URL** is required by Azure, set it to the domain name of your application (Ex: <https://<your-app-domain>/>).

The **Entity ID** is what Anark Collaborate calls the **Issuer ID**. Typically it is set to the domain name of your application.

If a user logs into Azure by navigating to Anark Collaborate, the **Reply URL** is where Azure will send a SAML response. Once Anark Collaborate received the SAML response, it will finish setting up the user's session. It must be set to <https://<your-app-domain>/samlreturn>.

LOGGING IN

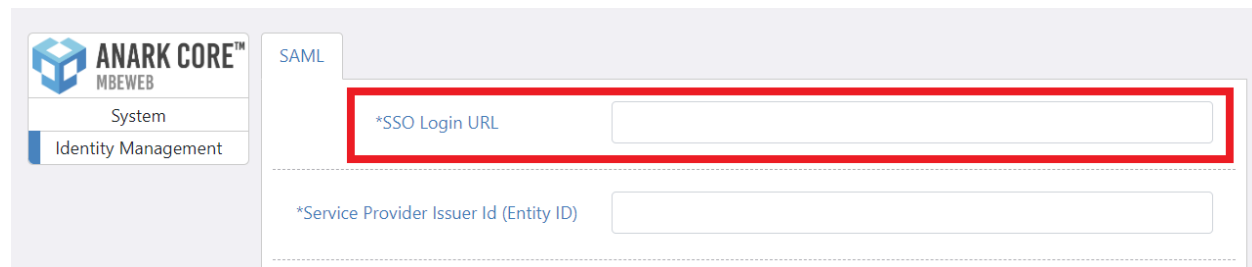
To set up Anark Collaborate for SSO, the first thing to do is find the **SSO Login URL**. In Azure, this can be found towards the bottom of the page, in the 4th section, titled "Set up".



The screenshot shows the Azure portal interface for configuring SSO. On the left, a navigation menu includes 'Roles and administrators', 'Users and groups', 'Single sign-on', 'Provisioning', 'Self-service', and 'Custom security attributes (preview)'. The main content area is titled 'Set up Anark Cloud SAML SSO Dev2' and contains the instruction: 'You'll need to configure the application to link with Azure AD.' Below this, a table lists configuration fields:

Login URL	https://login.microsoftonline.com/f3dd6f0d-b0d8-...
Azure AD Identifier	https://sts.windows.net/f3dd6f0d-b0d8-4caa-8b2...
Logout URL	https://login.microsoftonline.com/f3dd6f0d-b0d8-...

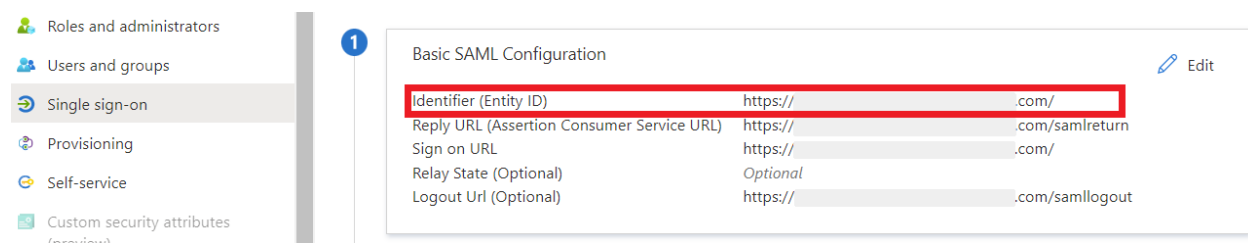
In Anark Collaborate, paste this value in to the "SSO Login URL" field. SAML request is sent using HTTP-Redirect binding.



The screenshot shows the Anark Collaborate SAML configuration page. On the left, the 'ANARK CORE™ MBEWEB' logo is visible, along with 'System' and 'Identity Management' tabs. The main content area is titled 'SAML' and contains two input fields:

- *SSO Login URL (highlighted with a red box)
- *Service Provider Issuer Id (Entity ID)

The next thing to paste into Anark Collaborate is the **Issuer ID**. As mentioned above, this can be found in the first section titled "Basic SAML Configuration".



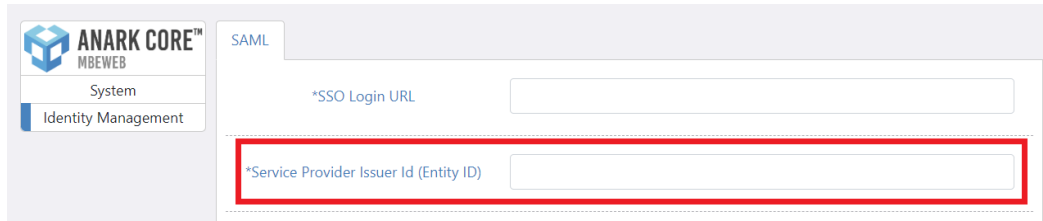
The screenshot shows the Azure portal interface for 'Basic SAML Configuration'. On the left, the navigation menu is the same as in the previous screenshot. The main content area is titled 'Basic SAML Configuration' and contains an 'Edit' button. Below this, a table lists configuration fields:

Identifier (Entity ID)	https://.../...
Reply URL (Assertion Consumer Service URL)	https://.../samlreturn
Sign on URL	https://.../
Relay State (Optional)	Optional
Logout Url (Optional)	https://.../samllogout



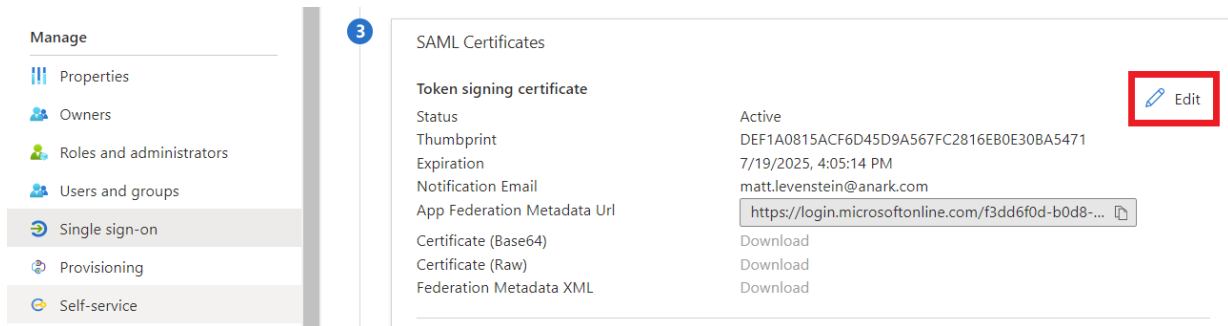
Anark Collaborate Deployment

Paste this value into “Service Provider Issuer Id (Entity ID)” field.



The screenshot shows the ANARK CORE™ SAML configuration interface. On the left, there is a sidebar with the ANARK CORE™ logo and 'MBEWEB' text, and a menu with 'System' and 'Identity Management' options. The main area is titled 'SAML' and contains two input fields. The top field is labeled '*SSO Login URL' and is empty. The bottom field is labeled '*Service Provider Issuer Id (Entity ID)' and is also empty; this field is highlighted with a red rectangular border.

The next thing to do is to find the **X.509 Public Certificate** in Azure. This can be found in the 3rd section of the application home screen in the Azure portal, titled “SAML Certificates”. Click “Edit” in the upper right corner.

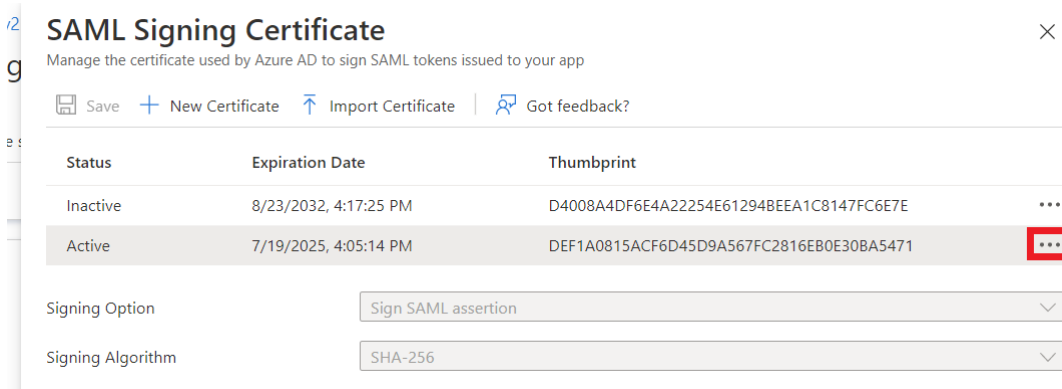


The screenshot shows the Azure portal interface for 'SAML Certificates'. On the left, there is a 'Manage' sidebar with options: Properties, Owners, Roles and administrators, Users and groups, Single sign-on, Provisioning, and Self-service. The main content area is titled 'SAML Certificates' and shows a table with one entry. The entry is for a 'Token signing certificate' with the following details:

Property	Value
Status	Active
Thumbprint	DEF1A0815ACF6D45D9A567FC2816EB0E30BA5471
Expiration	7/19/2025, 4:05:14 PM
Notification Email	matt.levenstein@anark.com
App Federation Metadata Url	https://login.microsoftonline.com/f3dd6f0d-b0d8-...
Certificate (Base64)	Download
Certificate (Raw)	Download
Federation Metadata XML	Download

An 'Edit' button with a pencil icon is located in the upper right corner of the certificate details, highlighted with a red box.

Find the certificate that is in the “Active” state and click the three dots on the right.



The screenshot shows the 'SAML Signing Certificate' dialog box in the Azure portal. The title is 'SAML Signing Certificate' and the subtitle is 'Manage the certificate used by Azure AD to sign SAML tokens issued to your app'. There are buttons for 'Save', '+ New Certificate', '↑ Import Certificate', and 'Got feedback?'. Below is a table with columns for 'Status', 'Expiration Date', and 'Thumbprint':

Status	Expiration Date	Thumbprint	
Inactive	8/23/2032, 4:17:25 PM	D4008A4DF6E4A22254E61294BEEA1C8147FC6E7E	...
Active	7/19/2025, 4:05:14 PM	DEF1A0815ACF6D45D9A567FC2816EB0E30BA5471	...

The three dots menu icon for the 'Active' certificate is highlighted with a red box. Below the table, there are two dropdown menus: 'Signing Option' set to 'Sign SAML assertion' and 'Signing Algorithm' set to 'SHA-256'.

Now click “Base64 certificate download”.



Anark Collaborate Deployment

SAML Signing Certificate

Manage the certificate used by Azure AD to sign SAML tokens issued to your app

Save + New Certificate ↑ Import Certificate | Got feedback?

Status	Expiration Date	Thumbprint
Inactive	8/23/2032, 4:17:25 PM	D4008A4DF6E4A22254E61294BEEA1C8147FC6E7E
Active	7/19/2025, 4:05:14 PM	DEF1A0815ACF6D45D9A567FC2816

Signing Option: Sign SAML assertion

Signing Algorithm: SHA-256

Notification Email Addresses: matt.levenstein@anark.com

- Make certificate active
- Base64 certificate download
- PEM certificate download
- Raw certificate download
- Download federated certificate XML
- Delete Certificate

Once downloaded, open in a text editor, and copy the text directly into “Identity Provider’s X.509 certificate” field.

ANARK CORE™
MBEWEB

System
Identity Management

SAML

*SSO Login URL

*Service Provider Issuer Id (Entity ID)

Identity Provider's X.509 certificate

ATTRIBUTE MAPPING

The last thing to do for a basic SSO set up with SAML, is provide an **attribute mapping**. An Attribute mapping maps IdP (Azure) SAML user attributes to Anark Collaborate internal user account properties. There are 4 required attributes: userid, groups, name, and email.



Anark Collaborate Deployment

The “Source Attribute” is the name of the attribute in the SAML document that comes from the IdP that contains user account information. And the dropdown on the right is where you will select an Anark Collaborate user account property.

The screenshot shows a configuration window titled “*SAML Attribute Mapping”. At the top, there is a text input field labeled “Defined attribute mappings”. Below this, there is a “Source Attribute” input field on the left and a dropdown menu on the right. The dropdown menu is currently empty. At the bottom right of the configuration area, there is a blue button labeled “Add Attribute Mapping”.

The “Source Attribute” names can be found in the Azure portal in the section titled “Attributes & Claims”.

The screenshot shows the Azure portal interface for “Attributes & Claims”. On the left is a navigation pane with options: “Roles and administrators”, “Users and groups”, “Single sign-on”, “Provisioning”, “Self-service”, and “Custom security attributes (preview)”. The “Single sign-on” option is selected. The main content area shows a table of attributes and claims. An “Edit” button is visible in the top right corner of the table area.

Attribute Name	Value
surname	user.surname
emailaddress	user.mail
name	user.userprincipalname
roles	user.assignedroles
groups	user.groups
givenname	user.displayname
Unique User Identifier	user.userprincipalname

Additional claims

Claim name	Value	
givenname	user.displayname	...
groups	user.groups	...
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailadd...	user.mail	...
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name	user.userprincipalname	...
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/roles	user.assignedroles	...
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname	user.surname	...

The “Claim name” is what you will use as the “Source Attribute” in Anark Collaborate.



*SAML Attribute Mapping

Defined attribute mappings

http://schemas.xmlsoap.org/ws/2

email

Add Attribute Mapping

You must repeat this process for all required fields. Again those fields are userid, email, groups, and name.

GROUP MAPPING

The **groups** attribute may not be provided to the SAML response by the IdP (Azure) by default. If this is the case, an attribute must be added. In order to accomplish this in Azure, select “Add a group claim” within the “Attributes & Claims” section.

Attributes & Claims

+ Add new claim + Add a group claim Columns | Got feedback?

Required claim

Claim name	Value
Unique User Identifier (Name ID)	user.userprincipalname

Additional claims

Within the “Group Claims”, select which groups that the user belongs to are to be returned in the SAML response. Then, select “Advanced options”.

Group Claims

Manage the group claims used by Azure AD to populate SAML tokens issued to your app

Which groups associated with the user should be returned in the claim?

None

All groups

Security groups

Directory roles

Groups assigned to the application

Source attribute *

sAMAccountName

⚠ This source attribute only works for groups synchronized from an on-premises Active Directory using AAD Connect Sync 1.2.70.0 or above. [Learn More](#)

Advanced options



Anark Collaborate Deployment

Within the “Advanced options”, check the box that says “Customize the name of the group claim”, and then add a name as well as a namespace (this is optional). If you choose to use a namespace, the full “Source attribute” name for the **groups** attribute will `<namespace>/<name>`

Advanced options

Filter groups

Attribute to match

Match with

String

Customize the name of the group claim

Name (required)

groups

Namespace (optional)

http://schemas.xmlsoap.org/ws/2005/05/identity/claims

Emit groups as role claims ⓘ

Apply regex replace to groups claim content

Save

Once “Save” is clicked, you can navigate over to Anark Collaborate, and add an attribute mapping for **groups**.

*SAML Attribute Mapping

Defined attribute mappings

http://schemas.xmlsoap.org/ws/2

groups

Add Attribute Mapping

By themselves, **groups** are just strings returned in the SAML response. In order for a user’s group memberships to translate to Anark Collaborate roles, a role mapping should be configured for each relevant group.



The Role Mapping setting defines rules that are used for determining which users and groups get mapped to roles in Anark Collaborate. In the following example, the **Identity Key** represents a key from the JSON sent by the identity provider, and **Identity Value** represents the associated value to the JSON key. The SAML integration provided the following JSON object:

```
{
  "issuer": "https://sts.windows.net/f3dd6f0d-b0d8-4caa-8b23-d3d628a171d5/",
  "groups": [
    "anark.na",
    "anark.eu",
    "anark.za"
  ],
  "identity/claims/role": [
    "ACM_ADMIN",
    "anarkviewer "
  ]
}
```

Using the setup shown in the example below, all users and groups that have the anarkviewer role will be assigned the “viewer” role within Anark Collaborate.

Role Mapping Defined role mappings

Identity Key	Identity Value
<input type="text" value="identity/claims/roles"/>	<input type="text" value="anarkviewer"/>

Anark Collaborate System Roles

x ▼

REQUEST SIGNING

Optionally, Anark Collaborate can sign SAML requests, just like the IdP (Azure) can. In order to enable this feature, you must select “Sign SAML Request”. And once you have generated a private key and public certificate, you must paste the private key in plaintext in the “Key” field seen below. The private key must be a well formatted PEM file.

Sign SAML Request

Key

Signature algorithm **Options**

▼



Anark Collaborate Deployment

Once you've selected a Signature algorithm (sha256 is a common default for this purpose), you can navigate back over to the Azure portal to find the section titled "SAML Certificates". Click the second "Edit" button.

Token signing certificate		Edit
Status	Active	
Thumbprint	DEF1A0815ACF6D45D9A567FC2816EB0E30BA5471	
Expiration	7/19/2025, 4:05:14 PM	
Notification Email	matt.levenstein@anark.com	
App Federation Metadata Url	https://login.microsoftonline.com/f3dd6f0d-b0d8-...	
Certificate (Base64)	Download	
Certificate (Raw)	Download	
Federation Metadata XML	Download	

Verification certificates (optional) (Preview)		Edit
Required	Yes	
Active	1	
Expired	0	

Here you will be able to upload the public certificate, so Azure can verify requests that Anark Collaborate has signed. This public certificate must be a well formatted PEM file with a .cer extension.

[Learn more](#)

Require verification certificates

Allow requests signed with RSA-SHA1

[Upload certificate](#)

Thumbprint	Key Id	Start date
D4008A4DF6E4A22254E612...	823c5146-0f2c-4297...	8/26/2022, 4:17 PM

LOGGING OUT

Anark Collaborate also supports single logout or SLO, in addition to SSO. There are two types of SLO. The first is IdP-initiated, and the second is SP-initiated, both are supported. IdP-initiated means that a user logged out of Azure, whereas SP-initiated means a user logged out of Anark Collaborate. SLO is entirely optional.

In the Azure portal, find the section titled "Basic SAML Configuration". **Logout URL** is where Azure will send the SAML logout request in the case of IdP initiated logout or the SAML logout response in the case of SP initiated logout. This value must be set to <https://<your-app-domain>/samllogout>.



Anark Collaborate Deployment

1

Basic SAML Configuration Edit

Identifier (Entity ID)	https://[redacted].com/
Reply URL (Assertion Consumer Service URL)	https://[redacted].com/samlreturn
Sign on URL	https://[redacted].com/
Relay State (Optional)	Optional
Logout Url (Optional)	https://[redacted].com/samllogout

If SP-initiated logout is desired, find the section titled “Set up”. The **Logout URL** found in this section will need to be entered into Anark Collaborate.

4

Set up Anark Cloud SAML SSO Dev2

You'll need to configure the application to link with Azure AD.

Login URL	https://login.microsoftonline.com/f3dd6f0d-b0d8-...
Azure AD Identifier	https://sts.windows.net/f3dd6f0d-b0d8-4caa-8b2-...
Logout URL	https://login.microsoftonline.com/f3dd6f0d-b0d8-...

Paste that value into “Logout url” field.

Logout url

ADMINISTRATION

SAML is enabled by the presence of these preferences in Anark Collaborate. If the following four settings are specified in “Identity Management”, SAML will be enabled, and users will be redirected to the login screen of your IdP when they navigate to Anark Collaborate: **SSO Login URL**, **Issuer ID**, **X.509 Public Certificate**, and **Attribute Mapping**.

In order to disable SAML, you must select “Clear all”.

User Role Mapping

Defined role mappings

Source Group

Anark Core System Roles

Add Role Map

Clear all Save

Every time an SSO user is authenticated with Anark Collaborate, their information is either added to the database or the database is updated with the latest information from the SAML response, which includes system roles.



Anark Collaborate Deployment

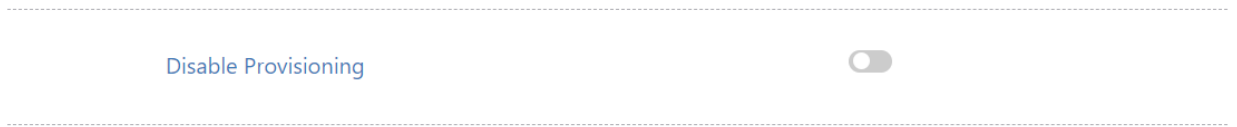
If SAML is enabled and an administrator would like to bypass the SAML login, and go directly to the local Anark Collaborate login screen, you can go to <https://<your-app-domain>/signin> and enter your administrator credentials.



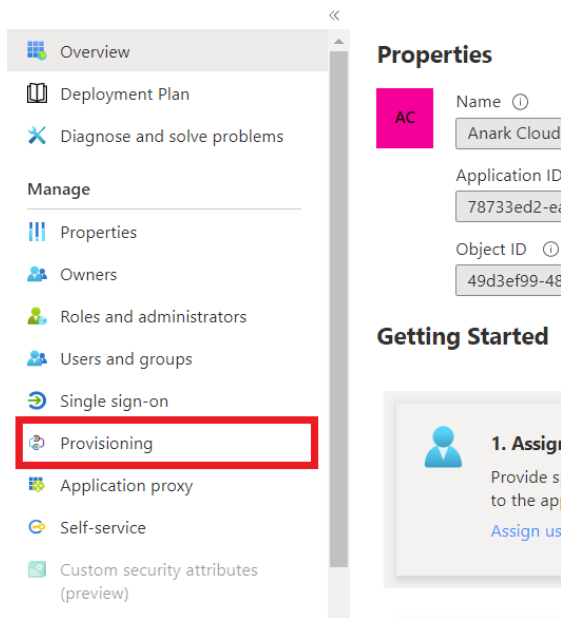
USER PROVISIONING

By default, users are provisioned on demand when they first login using SAML. This means that when a new user logs in, they are automatically added to our database and activated. If you would like more control over which users are allowed access, which users are active, and which users belong to what groups, Anark Collaborate supports the SCIM protocol. More information about SCIM can be found here: <https://www.simplecloud.info/>.

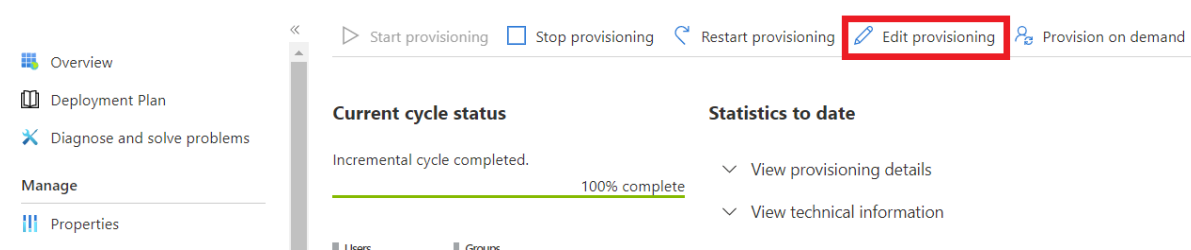
To use SCIM with Anark Collaborate, your IdP must support provisioning users with the SCIM protocol. In Anark Collaborate, you can toggle “Disable Provisioning”.



In the Azure portal, find the section titled “Provisioning” in the left-hand menu.



Then, select “Edit provisioning” to continue setup.




Anark Collaborate Deployment

Then select “Admin Credentials” in the Provisioning settings.

Provisioning ...

 Save  Discard

Provisioning Mode

Automatic 

Use Azure AD to manage the creation and synchronization of user accounts in Anark Cloud SAML SCIM based on user and group assignment.

Admin Credentials

Mappings

Settings

Provisioning Status ⓘ

On

Off

Inside of Admin Credentials, add the location of the Anark Collaborate SCIM API to the sectioned titled “Tenant URL”.

Provisioning ...

 Save  Discard

Provisioning Mode

Automatic 

Use Azure AD to manage the creation and synchronization of user accounts in Anark Cloud SAML SCIM based on user and group assignment.

Admin Credentials

Admin Credentials

Azure AD needs the following information to connect to Anark Cloud SAML SCIM's API and synchronize user data.

Tenant URL * ⓘ

https://[redacted]/api/scim/ 

Secret Token

Test Connection

Mappings

Settings




Anark Collaborate Deployment

After you're finished you can click "Test Connection" to make sure it works. "Secret Token" can be left blank. Next, click the "Mappings" section, and simply make sure Groups and Users are both enabled, as seen below.

Provisioning ...

 Save  Discard

Provisioning Mode

Automatic 

Use Azure AD to manage the creation and synchronization of user accounts in Anark Cloud SAML SCIM based on user and group assignment.

 Admin Credentials

 Mappings

Mappings

Mappings allow you to define how data should flow between Azure Active Directory and customappsso.

Name	Enabled
Provision Azure Active Directory Groups	Yes
Provision Azure Active Directory Users	Yes

Restore default mappings

If either Users or Groups isn't enabled, you can click into it and toggle the "Enabled" switch. Then click "Save".

Attribute Mapping ...

 Save  Discard

Name

Provision Azure Active Directory Users

Enabled

Yes No

Source Object

User

Source Object Scope

[All records](#)

Source Object

urn:ietf:params:scim:schemas:extension:enterprise:2.0:User

Target Object Actions

- Create
- Update
- Delete




Anark Collaborate Deployment

When you are finished with credentials and mappings, make sure “Provisioning Status” is set to “On”, and then click “Save”.

Provisioning ...

 Save  Discard

Provisioning Mode

Automatic 

Use Azure AD to manage the creation and synchronization of user accounts in Anark Cloud SAML SCIM based on user and group assignment.

∨ Admin Credentials

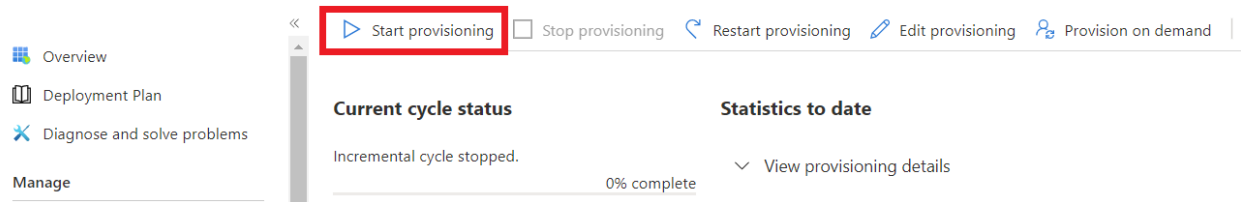
∨ Mappings

∨ Settings

Provisioning Status ⓘ

On Off

Once you are finished setting up SCIM, you can now click “Start Provisioning” in the main Provisioning menu.



The screenshot shows the Provisioning menu with a sidebar on the left containing 'Overview', 'Deployment Plan', 'Diagnose and solve problems', and 'Manage'. The main content area has a toolbar with 'Start provisioning' (highlighted with a red box), 'Stop provisioning', 'Restart provisioning', 'Edit provisioning', and 'Provision on demand'. Below the toolbar, there are two sections: 'Current cycle status' showing 'Incremental cycle stopped.' and '0% complete', and 'Statistics to date' with a 'View provisioning details' link.



ROLE MANAGEMENT

Anark Collaborate uses role based access control (RBAC) for content items and work instructions. Roles are created by an administrator and contain a set of application permissions and item access classifiers. Item access classifiers contain key-value pairs that will be used to identify which content items or work instructions the role has access to. The key-value pairs must match the content item key-value pair properties or work instruction key-value pair properties. Item access classifiers also contain privileges that enable or prevent downloading the matched content item.

For work instructions, a matching classifier gives access to the work instruction, and if item access inheritance is enabled, it will also give access to all content items within the work instruction.

A role can have multiple classifiers. When accessing an item, all user roles and classifiers will be treated as ORs, meaning only one of the classifiers have to match for access to be granted. The key-value pairs in a single classifier act as ANDs, meaning all key-value pairs must match the item for the classifier to match.

OVERVIEW

Anark Collaborate ships with default roles that have different application permission but no classifiers. It also supports defining custom roles so customers are not limited to the defaults.



Default roles cannot be edited or deleted from the system; however, they can be cloned to less the creation time of a new custom role. After creation, custom roles can also be cloned in addition to editing and deleting. The options are available when clicking the ellipsis icon on the bottom right of the role display card.

DEFINING CUSTOM ROLES

To create a new role, click on the “New” button (Desktop) or “Create” button (mobile) and then select “Role”. A modal will appear asking for a Role Name, Role Description (Optional), selected Application Permissions, and any Item Access Classifiers.

After giving the role a name, description, and selecting permissions, click on the “Add Access” button to see a new modal with options to enter key-value pairs matching the content item key-value pair properties, or work instruction key-value pair properties. Additionally, this modal gives toggles for enabling Download privileges to matching content items.



Anark Collaborate Deployment

< Create a Role

Role Name
Give your role a name

Role Description
Give your role a description

Application Permissions

- View Content Item**
Can open content items
- View Activity**
Can open activities
- View Work Item**
Can open work items
- View Group**
Can view groups
- Search Activity Group**
Can view activity group search results
- Search Content Item**
Can view content item search results
- Search User**
Can view user search results
- Search Activity**
Can view activity search results
- Search Group**

Execute Work Item
Can contribute by executing a work item

Create, Update, & Delete Content Item
Can author and manage content items

Create, Update, & Delete Template
Can author and manage templates

Create, Update, & Delete Work Item
Can author and manage work items

Report & Delete WI Execution
Can author execution reports and delete work item executions

System Administrator
Can view system logs, modify system preferences, manage roles, and manage identity integrations

All Content Items

If true, the role grants access to all content items with view and download privileges, irrespective of classifiers. For external access control, it should be enabled if you do not want to control download privilege based on classifiers. Administrators should use extra caution before enabling this option as a user with this role will have access to all content items if internal access control is enabled.

Item Access Classifiers
NO ITEMS [+ Add Access](#)

Cancel **Create**



<
Create Item Access Classifier

Item Classifier Name
Give your classifier a name

Metadata

Defined Properties

Key

Key

Value

Value

Type

String | v

Add Property

Content Item Privileges

View

Download

Cancel
Create

APPLICATION PERMISSIONS

The following table details each permission and what default role has this permission. NOTE: These permissions do not over-rule any Access Control specifications for given content.

Permission	Description	Default Roles with Permission
View Content Item	Can open published content items (this is separate from access control).	Collaborator, Content Author, Viewer, Work Instruction Executor, Work Instruction Manager, Work Instruction Author
View Activity	Can open an activity and view conversations and comments.	Collaborator, Viewer



View Work Instruction	Can open work instructions.	Viewer, Work Instruction Author, Work Instruction Executor, Work Instruction Manager
View Group	Can view group information.	Collaborator, Viewer
Search Activity Group	Can view activity group search results.	Admin, Collaborator, Viewer
Search Content Item	Can view content item search results.	Admin, Collaborator, Content Author, Viewer, Work Instruction Author
Search User	Can view user search results.	Admin, Collaborator, Viewer, Work Instruction Author
Search Activity	Can view activity search results.	Admin, Collaborator, Viewer
Search Group	Can view group search results.	Admin
Search Work Instruction	Can view work instruction search results.	Viewer, Work Instruction Author, Collaborator, Work Instruction Executor, Work Instruction Manager, Admin
Archived Item	Can view archived items.	Collaborator, Viewer, Content Author
Create, Update, & Delete Conversation	Can contribute through the creation and management of conversations.	Collaborator, Viewer
Create, Update, & Delete Comment	Can contribute through the creation and management of comments.	Collaborator, Viewer
Create, Update, & Delete Activity	Can contribute through the creation and management of activities.	Admin, Collaborator
Create, Update, & Delete Activity Group	Can contribute through the creation and management of activity groups.	Admin



Create, Update, & Delete User File	Can contribute through the creation and management of user files (upload to My Files).	Collaborator
Create, Update, & Delete Activity File	Can contribute through the creation and management of user files (upload to Activity and Share a Copy from My Files).	Collaborator
Create, Update, & Delete Work Instruction	Can author and manage work instructions.	Admin, Work Instruction Author
Execute Work Instruction	Can contribute by executing a work instruction.	Work Instruction Executor
Create, Update, & Delete Content Item	Can author and manage content items.	Content Author
Create, Update, & Delete Template	Can author and manage templates.	Content Author
Create, Update, & Delete User	Can contribute through the creation and management of users.	Admin
Create, Update, & Delete Group	Can contribute through the creation and management of groups.	Admin
Report & Delete WI Execution	Can author execution reports and delete work instruction executions.	Work Instruction Manager
System Administrator	Can view system logs, modify system preferences, manage roles, and manage identity integrations.	Admin

TROUBLESHOOTING AND LOG FILES

Anark Collaborate system creates different log files that can be used for diagnosing issues. Default log locations are shown below.



Anark Collaborate Deployment

- Application Server log is available at “/var/log/collab.log”.
- Web Server log is available at “/var/log/nginx/error.log”. If access logging is enabled, it is available at “/var/log/nginx/access.log”.
- Database Server log is available at “/var/log/mongodb/mongod.log”.
- Redis Server log is available at “/var/log/redis/redis.log”.

HIGH AVAILABILITY

Various techniques can be used to deploy Anark Collaborate server components to decrease downtime and to eliminate single points of failure. The below sections describe the supported mechanisms to support high availability.

WEB SERVER

A hardware or software load balancer (ex: NGINX as reverse proxy) can be used to ensure high availability. Load balancers can perform health checks and intelligently route the client request to only servers that are online.

NGINX plus (commercial version) servers can be configured into a high-availability cluster and supports a high availability solution based on “keepalived” (a routing software). For more information on this, please refer to the below link.

<https://docs.nginx.com/nginx/admin-guide/high-availability/ha-keepalived/>

APPLICATION SERVER

Anark Collaborate uses the NGINX (web server) load balancer to route the requests to the application servers running Node.js. There are multiple instances of Node processes running on the server to ensure high availability which are managed by the “PM2” process manager. The process manager is configured to restart the Node.js process automatically in the unlikely event of a crash.

DATABASE SERVER

To provide redundancy and high availability, replication needs to be enabled between MongoDB servers. Please refer to the below link for detailed help on configuring replica sets or follow the listed hints for a default configuration.

<https://docs.mongodb.com/manual/administration/replica-sets/>

Helpful hints:

- 1) Before configuring the replication, give the existing MongoDB “admin” user “clusterAdmin” role by using `db.grantRolesToUser()` method in Mongo shell.



- 2) Generate a key to enforce access control on a replica set and copy it to all servers
<https://docs.mongodb.com/manual/tutorial/deploy-replica-set-with-keyfile-access-control/#deploy-repl-set-with-auth>
- 3) Edit MongoDB config file (/etc/mongod.conf) to update or add the below settings and restart MongoDB server.

```
bindIp: 127.0.0.1,192.168.0.50 // Add IP address so that other servers can connect to it
unixDomainSocket: // Disable UNIX socket
enabled: false
security:
authorization: enabled
keyFile: /var/db/mongodb/replicaauth.key // Key file that was created in step 2
replication:
replSetName: "collab-rs0" // Replication set name
```

- 4) Configure MongoDB replication using Mongo shell (rs.initiate(), rs.add(), rs.conf(), rs.status()). If you want to add the existing QA standalone MongoDB server to replica set, add it first and then add the other servers.
- 5) After configuring the replica servers, enter the connection string (example shown below) in pm2.json file for "DB" setting.

```
"DB":"mongodb://collab:collab@192.168.0.50:27017,192.168.0.51:27017,192.168.0.53:27017/anarkmbedev?replicaSet=collab-rs0"
```

- 6) Restart Node processes by using "pm2 stop all" and "pm2 start /usr/local/collab/pm2.json".
- 7) If there are no errors in the log file with respect to MongoDB connections, save the PM2 startup script as described in step 15 in the Installation Instructions (previous section).

REDIS SERVER

Redis provides Redis Sentinel for high availability. Please refer to the below link for more information, otherwise follow these hints for configuring.

<https://redis.io/topics/sentinel>

Helpful hints:

- 1) Edit the **redis.conf** file (/etc/redis.conf)
bind 127.0.0.1 192.168.0.50 // Add the IP address so that other instances can connect
#unixsocket /tmp/redis.sock // disable UNIX socket
#unixsocketperm 770
replicaof 192.168.0.50 6379 // add this only in replica nodes, here 192.168.0.50 is the master IP
- 2) Edit **redis-sentinel.conf** file (/etc/redis-sentinel.conf). For master-name, use "mbewebmaster".
bind 127.0.0.1 192.168.0.50 // Add the IP address
logfile "/var/log/redis/sentinel.log"
sentinel monitor mbewebmaster 192.168.0.50 6379 2
sentinel down-after-milliseconds mbewebmaster 30000
sentinel parallel-syncs mbewebmaster 1
sentinel failover-timeout mbewebmaster 180000



Anark Collaborate Deployment

- 3) Use systemctl enable command to start Sentinel at boot time.
- 4) Enter the connection string in pm2.json for REDIS setting (example shown below)

```
"REDIS": "192.168.0.50:26379,192.168.0.51:26379,192.168.0.53:26379"
```

- 5) Restart Node processes by using “pm2 stop all” and “pm2 start /usr/local/collab/pm2.json”
- 6) If no errors in the log file with respect to Redis connections, save the PM2 startup script as described in step 15 in the Installation Instructions above.



Anark Collaborate Deployment

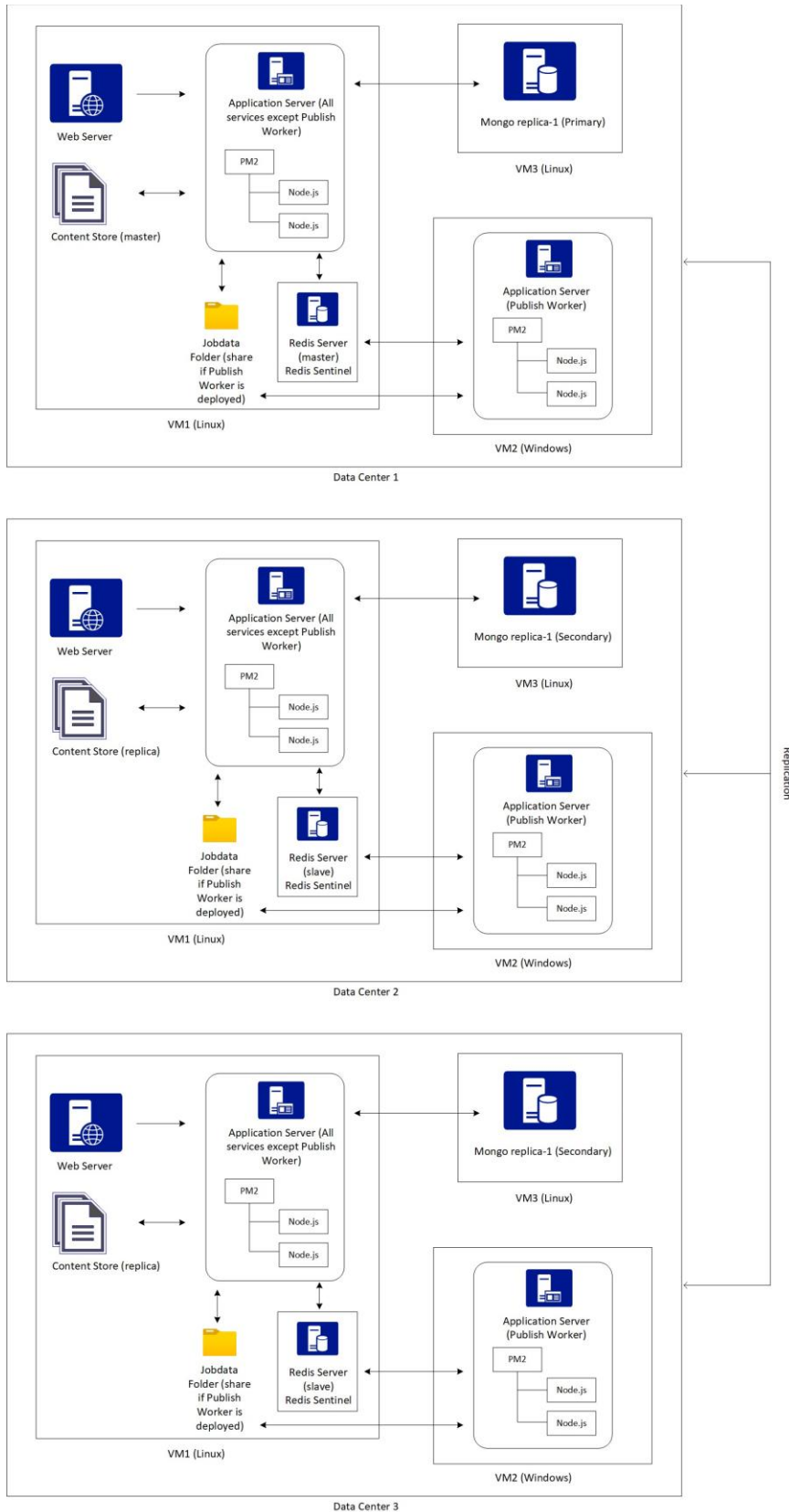


Figure 2: An example of multi-site, highly-available Anark Collaborate deployment.



PERFORMANCE/SCALING RECOMMENDATIONS

WEB SERVER

NGINX can run multiple worker processes. Each process is capable of handling a large number of simultaneous connections. By running more worker processes, NGINX should be able to handle more Requests per Second (RPS), Connections per Second (CPS), and higher Throughput. The below link has benchmark information on how the number of CPU's allocated to NGINX can impact performance.

<https://www.nginx.com/blog/testing-the-performance-of-nginx-and-nginx-plus-web-servers/>

NGINX directives "worker_processes" and "worker_connections" can be used to specify the optimal values for the number of worker processes and the number of connections that each process can handle.

A few Linux settings can impact performance as well. Please refer to step 18 in the "Installation Instructions" section for more information on these settings.

Disk I/O should be monitored in cases of slow performance. Typically, OS caches the frequently used files in "page cache" if there is enough RAM to store the dataset. If the working data set can fit into the RAM, then NGINX can offer better performance. Also, network performance can create performance issues, so content should be stored close to the user for optimal performance.

Logging every request in the access log consumes both CPU and I/O cycles. It is recommended to turn off access logging or enable buffer if access logging is enabled to improve performance.

There are limits to performance improvements achieved with vertical scaling, if adding more resources to the same server and utilizing those does not provide the required performance, horizontal scaling strategy should be chosen by adding more web servers behind a load balancer.

APPLICATION SERVER

The PM2 process manager is used to manage the Node processes in the application server. By default, two Node processes run in application server. If proxy timeout is reported by the web server on high load, PM2 can be configured to run more Node processes to handle more load based on the number of cores available on the server.

The application server can be also deployed to multiple server boxes. The NGINX load balancer can be easily configured for this setup to achieve optimal performance.

DATABASE SERVER

The storage engine used by MongoDB is multi-threaded, so number of available CPUs can impact performance. According to MongoDB docs, "Throughput INCREASES as the number of concurrent active operations increases up to the number of CPUs". The "mongostat" utility provides statistics on the number of active reads/writes in the (ar|aw) column and optimum number of concurrent active operations can be determined by experimenting and measuring throughput.

MongoDB provides the best performance if the working set (data that clients access most often) fits into memory. The MongoDB storage engine allocates a specific amount of memory (RAM) to be used for storing internal cache



and uses the rest of the available memory that is not used by other processes via the filesystem cache (used by the OS to reduce disk I/O). Working set sizes larger than the system's RAM stress the I/O capacity of disk drives.

For debugging disk I/O related performance issue, run the "serverStatus" command and analyze the information for "wiredTiger.cache.pages read into cache" and "wiredTiger.cache.pages written from cache" output fields. These can provide an overview of the I/O activity. Also, a high value for "wiredTiger.cache.unmodified pages evicted" field indicates that working set does not fit in memory. Disk reads should be monitored and if there is lot of activity, it is also a good indication that working set does not fit in memory.

A few Linux settings can impact performance as well. Please refer to step 18 in the "Installation Instructions" section for more information on these settings.

If there are limits on available hardware configurations for running the database server (ex: max RAM available is 64 GB) and very large data sets needs to be supported, MongoDB supports sharding. Using sharding, data is distributed across multiple machines and it can support deployments with very large data sets and high throughput operations.



SECURITY GUIDELINES

Listed below are some recommended security guidelines for Anark Collaborate deployment.

- Verify that the VM or bare metal where Anark Collaborate will be deployed has all the latest RHEL security updates, otherwise it is recommended to install the latest updates prior to Anark Collaborate deployment. Use key-based authentication and disable password authentication for SSH.
- Create a service user account to run Anark Collaborate services. Set service account default shell to `/dev/null` to harden against attackers gaining system access.
- Disable read and write access to Anark Collaborate content store volume for any users other than the service user account that is being used.
- Encrypt all the passwords (DB, Redis, Content replication user) that is specified in the `PM2.json` configuration file.
- Use HTTPS for secure communication, NGINX (web server) should be configured to use TLS 1.2 or newer. Use the Strict-Transport-Security HTTP header to ensure that browser only connects to Anark Collaborate over HTTPS.
- Enable secure cookies, this is controlled by `"ENABLETLS"` or if `NODE_ENV` is set to `"production"` Anark Collaborate config setting.
- Set the session timeout to meet your security needs, by default, it is set to 2 hours.
- Use the Content-Security-Policy header to mitigate cross site scripting (XSS) attacks.
- Specify the X-Content-Type-Options header to prevent MIME sniffing.
- Modify the Cache-Control header properties if content item resources should not be stored in browser cache. Please note that this might impact performance when viewing a content item.
- Anark Collaborate apps cannot be embedded in a page from a different origin by default. Use Content-Security-Policy headers `"frame-ancestors"` attribute to specify the allowed origins.
- Enable CORS using the required headers in NGINX if Anark Collaborate apps needs to be accessed from a different origin.

SYSTEMS INTEGRATION

AUTHENTICATION

Out of the box, Anark Collaborate supports local authentication and SAML authentication. It can be customized and configured to be integrated with the single sign-on solution that is deployed in the enterprise (SAML 2.0, OpenID Connect) or perform authentication against LDAP server or other systems if required.

Another option is to deploy Anark Collaborate behind a reverse proxy server running an SSO agent. The reverse proxy server performs the SSO authentication and forwards the required information about the user to Anark Collaborate. Additionally, Anark Collaborate can be customized to enable multi-factor authentication (MFA).

CONTENT ACCESS CONTROL (AUTHORIZATION)

Content access in Anark Collaborate can be managed by an external system (ex: PLM). This is useful when content access in Anark Collaborate needs to be enforced based on the access control rules specified in an external system. Before any user can view or collaborate with a specific content, an external system web service is invoked by Anark



Anark Collaborate Deployment

Collaborate to check whether the user has access to the content. Based on the response of the external system web service, a user will be granted or denied content access.

Content item search can support external access checks. When a user performs a search, only content items that they have access (external web service will be invoked to check the access) is shown in the search results.

ACCESSING EXTERNAL USERS AND GROUPS

Anark Collaborate can be customized and configured to access users and groups from an external system. This eliminates the admin overhead of manually adding users and Activity Groups in Anark Collaborate. When creating an Activity for collaboration in Anark Collaborate, integration allows the Activity Owner to access the external users and Activity Groups.

To access users or groups, Anark Collaborate can run a background task periodically to get the users and groups from an external system. This gets cached in Anark Collaborate for faster search and keeps the existing users in Anark Collaborate database up to date. Another option is to invoke a web service to perform a search query for users or groups in the external system when the Activity Owner searches for users or Activity Groups in Anark Collaborate.

It is also possible to use the user roles from an external system. Rules can be set up in the integration to map the user role(s) from an external system to Anark Collaborate user role(s). For example, if external system users with a “Contributor” role needs to create activities in Anark Collaborate which requires “Collaborator” role a rule can be setup in the integration to do the mapping.



LICENSING

The Anark Collaborate Software and Software licensed for use with Anark Collaborate are licensed per process for the system, and additionally per named user for user access licenses. The Anark Collaborate Software includes APIs which may be used by the licensee in derivative works solely to extend the capabilities of Anark Collaborate or to integrate the Anark Collaborate Software with other software systems. Please see EULA for more information.

NAMED USER

A “Named User” is a user of the Software, where the license is deemed available to only the specified user at any given time. Anark Collaborate admin can check the named user licenses usage by clicking on the **Help** link on the navbar.

