ArchiveKeeper

Setup Guide

Version: 12/2024/1.0

COMPRISE GmbH

www.comprise-world.com | archivekeeper-support@comprise-world.com

COMPRISE
MADE IN EUROPE

Content

1. General 2
2. Setup AWS components 2
2.1 Setup other AWS components 2
2.1.1 Postgres Database 2
2.1.2 OAUTH/OIDC 2
2.2 Launch EC2 instance of ArchiveKeeper 4
2.3 Make the Backend and UI accessible from the internet 5
3. Start ArchiveKeeper 5
3.1 Configuration 5
3.1.1 Postgres Database 5
3.1.2 Mounting file storage 6
3.1.3 OAuth/OIDC 6
3.1.4 UI - Backend connection 6
3.2 Deactivating the UI 7
3.3 Start containers 7
4. Configuration Property-List 8

1. General

This guide describes what AWS components are needed to run the ArchiveKeeper (AK) AMI in AWS, how to configure AK and how to start it.
ArchiveKeeper consists of three applications. The backend with a REST-API (ak-be), an angular frontend (ak-ui) and the key management service (ak-kms). The applications run as docker containers and can be started via a docker-compose config.

This guide is **for Versions 2.5.0-1.0 and greater**. The version number is built from the version of ArchiveKeeper, a hyphen as delimiter and the version for the AMI itself.

2. Setup AWS components

You need to set up the following to use the AMI.

- Postgres Database (e.g. RDS)
- OAUTH/OIDC (e.g. Cognito User Pool)
    - Lambda for token transformation
- File Storage (e.g. EFS)

You only need to have the following components if ArchiveKeeper should be reachable from outside of AWS.

- Hosted zone (Route 53)
- Certificate (AWS Certificate Manager)
- Load balancers (EC2)

If ArchiveKeeper is only reachable from inside the AWS or you want to use it just via the REST API, the UI application can be turned off. See section Deactivating the UI for more information.

2.1 Setup other AWS components

This section describes how to set up some of the components via AWS services or how to prepare them to use them together with ArchiveKeeper if you already have them running.

2.1.1 *Postgres Database*

If you don't have a Postgres-compatible database yet you can create one via Amazon RDS e.g. a PostgreSQL database with an engine version >= 15.

2.1.2 *OAUTH/OIDC*

If you don't have a user management yet, go to Amazon Cognito and create a user pool. The pool should have the user attributes email, given name and family name.

2.1.2.1 Branding

We recommend adjusting the branding to your corporate design if you use the ArchiveKeeper UI.
If you don’t want to use a generic domain, you can use create a custom domain. For this you need a

certificate, in the section Make UI accessible - Certificate you can read more about certificates in AWS.

2.1.2.2 App Client

Create an app client for ArchiveKeeper. If you have an existing user pool, you can do so in the submenu *App clients*. If you are creating a new user pool you will be asked for the client name in the creation process.
In the *App client*, you can configure the allowed callback URLs in the *Login pages* tab, these are the URLs that the user can be redirected to after authentication. The redirect URL (*oidc.redirectUri*) from ak-ui (see OAuth/OIDC) must be one of the defined callback URLs.
Besides the callback URL, the OAuth 2.0 grant types need to include *Authorization code grant*. The OpenID Connect scopes need to include *Email*, *OpenID* and *Profile*.
After creating the app client, the client id needs to be added to the configuration of ak-be and ak-ui (See section Start ArchiveKeeper → Configuration). The client id can be found in the *App client list* of the user pool after creating the app client.
If you adjust an existing app client instead of creating a new one don't forget to create a domain if the app client doesn't have one yet.

2.1.2.3 How to configure the roles of a user

In AK the roles of a user define what permissions the user has. There is only 1 predefined role, the role *ADMIN*. You can define additional roles if you want.
Every AK-role needs to have a group in the user pool. The group name needs to be in the form p_[Role name].
To assign a role to a user add the user to the group representing the role.
If you don't create a group p_ADMIN and add at least 1 user to the group, you won't be able to use ArchiveKeeper. You can find more information about roles and permissions in the user manual or the API documentation.

2.1.2.4 Customize the access token claims

The user pool needs to have lambda trigger to customize the access token claims. This is needed to bring the token in the form expected by ak-be.
You can create the lambda function in the connection dialog. To connect the lambda and the user pool go to *Extensions* and press *Add Lambda trigger*. Choose the trigger type *Authentication* with *Pre token generation trigger*.
Furthermore, select trigger event version Basic features + access token customization and create a new lambda in the section lambda function.
The lambda can have the default settings (runtime etc.) and should use the following javascript code:

```
exports.handler = async (event) => {
  var claims = {};

  claims["aud"] = event.callerContext.clientId;
  claims["resource_access"] = {};
  claims["resource_access"][event.callerContext.clientId] = {};
  claims["resource_access"][event.callerContext.clientId]["roles"] = [];

  var groups = event.request["groupConfiguration"]["groupsToOverride"];

  if (groups && groups.length > 0) {
    claims["resource_access"][event.callerContext.clientId]["roles"] = groups;
  }

  if ("email" in event.request["userAttributes"]) {
     claims["email"] = event.request["userAttributes"]["email"];
  }
  if ("given_name" in event.request["userAttributes"]) {
     claims["given_name"] = event.request["userAttributes"]["given_name"];
  }
  if ("family_name" in event.request["userAttributes"]) {
     claims["family_name"] = event.request["userAttributes"]["family_name"];
  }

  event.response = {
    "claimsAndScopeOverrideDetails": {
      "accessTokenGeneration": {
        "claimsToAddOrOverride": claims
      }
    }
  };

  return event;
};
```

Codeblock 1 Token Transformer

The code does the following:

- It adds an aud claim that holds the clientId of the app client
- It adds the groups of the user as resource_access
- Add claims for the email, given name and family name of the user.

2.2 Launch EC2 instance of ArchiveKeeper

Select an instance type depending on the suspected load for ArchiveKeeper. If you chose the wrong type, you can adjust it later but need to stop the EC2 instance.
For the Network settings you can choose the security groups depending on your preferences. You should decide if the ArchiveKeeper can be accessed by everyone in your VPC, the internet or only from chosen services or IP addresses. However, you should make sure that the EC2 instance can send requests to the Database and that the instance can reach the remote file storage (see Mounting file storage).
You can keep the default storage (volume) settings unless you plan to not use a remote storage for your archived files. In that case you need to adjust the storage to the expected size of the uploaded files. We advise against using the local storage.

2.3 Make the Backend and UI accessible from the internet

2.3.1.1 Application Load Balancer

Create or update an internet-facing application load balancer.
Add one listener with 2 rules. One for accessing the backend and one for accessing the UI. We recommend listening to HTTPS connections on port 443.
The rule for the backend should match the path pattern */api/** and have a target group with the target type instances. The target group should use the AK EC2 instance as the registered target with port 8080 and use the HTTP protocol.
For the health check the path */manage/health* and the success codes 200-299 can be used.

For the UI the default rule can be used. It should redirect all other requests to the target group of the UI. This target group should also have the target type instances and use the ArchiveKeeper EC2 instance as the registered target with port 4200 and the HTTP protocol. For the health check the path */health* and the success codes 200-299 can be used.
The load balancer can use an SSL/TLS server certificate from ACM if you don't have a certificate yet. See the Certificate section for more information.

2.3.1.2 Certificate

If you don't have a certificate, you can use the AWS Certificate Manager (ACM) to create one. First request a public certificate with a fully qualified domain name which uses a domain name that fits your hosted zone. In other words, that ends with the domain of your hosted zone.
This will create a certificate in status pending validation. To validate it press *Create records in Route 53* and finish the dialog that is opened. After 1 or 2 minutes the certificate will change to issued and a record in your hosted zone has been created.

3. Start ArchiveKeeper

3.1 Configuration

Each application has a configuration file. For ak-be and ak-kms this is an environment file that can be found at ak-be/env and kms/env.
The UI is configured via the json file at ui/env-settings.json.

3.1.1 *Postgres Database*

The AK uses a Postgres database to store all data except the archived files. ak-be and ak-kms set up the needed tables by themself. However, they need to know which database to use. The following properties need to be set for ak-be and ak-kms:

Property	Description
spring.datasource.url	The URL, including the protocol (jdbc:PostgreSQL), that the applications should use to access the DB. E.g. jdbc:postgresql://[Name].[Random_Hex].[AWS_Region].rds.amazonaws.com/[Database]?currentSchema=ArchiveKeeper The schema is created automatically, but the given [Database] needs to exist in the database server
spring.flyway.schemas	The schema where the tables of the application should be placed. e.g. ArchiveKeeper. Note the schema

Property	Description
	in **spring.datasource.url** and **spring.flyway.schemas** need to match.
spring.datasource.username	Name of the DB user
spring.datasource.password	Password of the DB user

ak-be and ak-kms can use different or the same database servers and databases. However, they need to use different schemas.

3.1.2 *Mounting file storage*

The storage for the documents is injected in the backend container as persisted volume. By default, the configuration uses the folder "/storage/archive-keeper" of the EC2 instance for the volume. To be independent of the EC2 instance we recommend using Amazon EFS (or similar) and mounting it in the "/storage/archive-keeper" path of the EC2 instance. ak-be will automatically create the needed folders in the path on startup. If you use another path, don't forget to adjust "at.risedev.ArchiveKeeper.rootfolder" in the ak-be environment file.

3.1.3 *OAuth/OIDC*

The Users authenticate themselves via an access token in the REST requests against ak-be. Therefore, ak-be needs the clientId to validate the token and get the roles the user has in ArchiveKeeper.

Property	Description
at.risedev.ArchiveKeeper.idm.client-id	The clientId defined in the authorization server
at.risedev.ArchiveKeeper.idm.default-issuer	The URL of the authorization server e.g https://cognito-idp.[Region].amazonaws.com/[User_pool_ID]

The UI redirects unauthenticated users to the authorization server therefore the following properties in the json file need to be set:

Property	Description
oidc.issuer	The URL of the authorization server e.g https://cognito-idp.[Region].amazonaws.com/[User_pool_ID]
oidc.clientId	The clientId defined in the authorization server
oidc.redirectUri	The URI, where the authorization server should redirect the user after a successful authentication. This should be the Url that users can use from outside AWS to access the UI.
oidc.postLogoutRedirectUri	The URI, where the authorization server should redirect the user after a logout. Often is the same as oidc.redirectUri.

ak-kms is only called by ak-be therefore it does not need any configuration for OIDC.

3.1.4 *UI - Backend connection*

The UI needs to know how to reach the backend, therefore set the property base in ui/env-settings.json. Note that the UI runs on the machine of the user so the URL needs to work from outside of AWS. It should also match the load balancer settings.

Property	Description
Base	The URL under which the ak-ui can reach the backend API. E.g. "https://your-domain.com/api"

3.2 Deactivating the UI

To deactivate the UI the compose.yaml file needs to be adjusted. Set the replicas of the ak-ui service to 0.

3.3 Start containers

After everything is configured, the applications can be started via docker compose.

4. Configuration Property-List

This list describes how the ArchiveKeeper backend can be configured.

Property	Description	Default Value	Possible Values/ Datatype
at.risedev.ArchiveKeeper.prune			
deletedDocumentCron	The cron-expression defines when the document prune jobs run.	-	Cron Expression
deletedDocSpaceCron	The cron-expression defines when the document space prune jobs run.	-	Cron Expression
deletedDocument	How long a document needs to be marked as (physical) deleted before it gets pruned.	P2D	`Duration`
deletedDocSpace	How long a document space needs to be marked as deleted before it gets pruned.	P2D	`Duration`
binaryCreated	Binaries that are longer in the state Created without being referenced by a document (revision) than the given value will be pruned by the document prune job.	PT1H	`Duration`
binaryUploading	Binaries that are longer in the state Uploading without being referenced by a document (revision) than the given value will be pruned by the document prune job.	P1D	`Duration`
tempDirectory	Files in the temp directory that have not been modified for the given duration will be deleted.	P1D	`Duration`
batchSize	How many documents get pruned in one batch.	1000	`int`
at.risedev.ArchiveKeeper.cache.config			
jwtToken.spec	Caffeine spec for jwtToken cache	"maximumSize=256, expireAfterWrite=600s"	

jwtAuth.spec	Caffeine spec for jwtAuth cache	"maximumSize=256, expireAfterWrite=180s"	
mdSchema.spec	Caffeine spec for mdSchema cache	"maximumSize=32, expireAfterWrite=300s"	
documentSpace.spec	Caffeine spec for documentSpace cache	"maximumSize=512, expireAfterWrite=5s"	
roles.spec	Caffeine spec for roles cache	"maximumSize=512, expireAfterWrite=5s"	
userPrettyName.spec	Caffeine spec for userPrettyName cache	"maximumSize=512, expireAfterWrite=600s"	
cipher.spec	Caffeine spec for cipher cache	"maximumSize=512, expireAfterWrite=60s"	
at.risedev.ArchiveKeeper.admin.nodeId	The id of the node running the application		

at.risedev.ArchiveKeeper .entityScan.packages	A list of package names to scan for database entities.	at.risedev.ArchiveKeeper	List of Strings
at.risedev.ArchiveKeeper. documentSpace	Default values for new or updated document spaces		
defaultOwnerGroup	This group gets read, writte and delete access on every new docspace. Empty means no default group.	default-owner-group	
hardLimit	The default hard size limit of a document spaces. Is applied if the user doesn't define a value in the create or update request. Value is in MB.	20000	Java long
softLimit	The default soft size limit of document spaces. Is applied if the user doesn't define a value in the create or update request. Value is in MB.	10000	Java long
maxUploadFileSize	The default maximum size limit of a document for the document space. Is applied if the user doesn't define a value in the create or update request. Value is in MB.	100	Java long
retentionDuration	The default retention period of a document for a space. Is applied if the user doesn't define a value in the create or update request.	P0DT0S	All positive ISO 8601 strings
at.risedev.ArchiveKeeper.privacy. userPrettyName	How the pretty name of a user is built. This pretty name is used in the front end to display what user did what action, e.g., who uploaded a document.	FIRST_LAST	FIRST_LAST, UUID, MAIL, USERNAME, NOT_SET
at.risedev.ArchiveKeeper.delete. autoDeleteFactor	The factor determines the maximum number of autodeleteable documents that can be marked as deleted in a single run of the prune document task.	1	float

at.risedev.ArchiveKeeper. rootFolder	The path of the root folder for the storage of the uploaded documents.	${System's temporary directory}/ArchiveKeeper	
at.risedev.ArchiveKeeper. tempFolder	The path of the folder where uploaded documents are temporarily stored.	${rootFolder}/temp	
at.risedev.ArchiveKeeper.security. cors.allowedOrigins	The list of origins that are allowed to send requests to the ArchiveKeeper backend. Allows patterns like *.		List of Strings
at.risedev.ArchiveKeeper. connectors.audit.database.enabled	Defines if audit records for user actions should be stored in the database. The audit records will be written in the log output if this is false.	false	boolean
at.risedev.ArchiveKeeper. connectors.diskstorage	These properties configure the file storage. This storage contains the files that are connected to documents.		
Location	The path of the storage location folder.	${rootFolder}/storage	
splitDocumentSpace	Defines if the folders for the document space are ordered in a tree structure or if they are all stored directly in the storage folder (see location). On true the document space id is split ${docSpaceDepth} parts of size ${docSpacePartSize} and for every part there will be one folder. In the final folder, there will be an additional folder containing the whole space id to ensure there are no collisions. This prevents the storage folder from having too many elements in it when there are a lot of document spaces.	false	boolean
v0Supported	Defines if a storage connector version 0 is used. The versions differ in where the files are stored.	true	boolean
v1Supported	Defines if a storage connector version 1 is used. The versions differ in where the files are stored.	false	boolean
docSpaceDepth	Defines how many subfolders (tree levels) are above the folder for the document space. Only applied if splitDocumentSpace is true. See splitDocumentSpace for more information.	1	int

docSpacePartSize	Defines how many characters of the document space id are used for the name of a subfolder. Only applied if splitDocumentSpace is true. See splitDocumentSpace for more information.	2	int
binaryLocationDepth	In contrast to the document space folders, the binaries are always stored in a tree structure. This configuration defines in how many subfolders the (binary) file is stored. It follows the same concept as described in splitDocumentSpace. If the value is 0 the file is directly stored in the folder for the document space. Not used in version 0 storage connector.	1	int
binaryLocationPartSize	In contrast to the document space folders, the binaries are always stored in a tree structure. Defines how many characters of the binary id are used for the name of a subfolder. It follows the same concept as described in splitDocumentSpace. If the value is 0 the file is directly stored in the folder for the document space. Only applied if splitDocumentSpace is true. Not used in version 0 storage connector.	2	int
storeMethod	Define how the uploaded files are moved from the temporary folder to the final storage. MOVE means that the file is just moved to the other location without reading it again. COPY means that the file is copied to the new location. WRITE means that the files are read by a stream and written to the new location.	WRITE	MOVE, COPY, WRITE
at.risedev.ArchiveKeeper.storage	These properties configure what storage connectors are used		
storageMigrationFactor	This factor defines how many binaries are migrated to the current write storage connector in one run of the migration job. The migration only runs if the value is > 0 and more than one storage connector is active.	1	float
writeStorageConnector	Defines which storage connector version is used as write connector. If this value is not set the connector with the highest version will be chosen.		internal, internal_v1
at.risedev.ArchiveKeeper.idm	These configurations affect how the user is authenticated and authorized		
Type	What type of authentication is used.	BASIC	BASIC, JWT

defaultIssuer	An identification of the default issuer. Is used for initial users and if a user is created without a defined issuer.	${spring.security.oauth2.resourceserver.jwt.issuer-uri} or ArchiveKeeper-internal	
defaultRoles	These roles are created if there no roles on startup of the application. A role has a name, description and a list of permissions. The permission value ALL gives all the role every permission.	name: ADMIN description: Administrator role with all permissions permissions: ALL	
clientId	The client id of this application. This is used to validate the token and find the application-specific claims.		
usernameClaimName	The name of the claim that holds username.	sub	iss, sub, aud, exp, nbf, iat, jti
firstnameClaimName	The name of the claim that holds the first name of the user.	`given_name`	
lastnameClaimName	The name of the claim that holds the last name of the user.	family_name	
emailClaimName	The name of the claim that holds the email of the user.	email	
acceptedAcr	If any accepted acr values are set the car claim in the token needs to have one of the defined values. If no accepted acr are defined the check is skipped.		List of Strings
createUserIfMissing	If a user that doesn't exist in the ArchiveKeeper database authenticates itself with a token it is created when the value is true. If it is false the authentication fails.	false	boolean

updateUserData	Defines if the user data in the ArchiveKeeper database should be updated when the user logs in and its user data has changed.	false	boolean
unsafeNormalizeRoles	Role names in a token are normalized by removing leading "p_" and transforming it into upper case. Only has an affect if a token is used for autnetication.	true (v2.5.0)	boolean
initialUser	The initial user is created when it is defined and no other user in the database exists. The initial user has the properties "name", "password" and roles. Where roles is a list of strings.		
at.risedev.ArchiveKeeper.preservation	The configuration for the revisionsafty		
Enabled	If the preservation job creating the revision safety evidence is active. If this is turned off no evidence is created. This means manipulation of the data through an atacker would not be noticed. If it it is turned on again all the document revisions that don't have evidence will get an evidence.	false	boolean
minAge	The minimum age a revision needs to have, that it is included in a timestamp.	PT1H	Duration
maxAge	If an document revision is older than the defined age a timestamp will always be created. This holds true even when there a less revisions then defined in minObjectsPerTimestamp.	PT2H	Duration
minObjectsPerTimestamp	The minum number of document revisions for which a timestamp will be created.	32 768	int
maxObjectsPerTimestamp	The maximum number of documents that can be included in a timestamp.	131 072	int
trustStore	The resource location of the trust store.	classpath:tsp-tsa-default.jks	Absolute or relative Path
trustStorePassword	The password of the trust store.	changeit	
checkRevocation	If it is checked certificates of timestamps were revoked.	true	boolean
certificateCheckInterval	How often the certificates should be checked if the were revoked. Only has an effect when checkRevocation is true.	PT24H	Duration

Cron	The cron-expression defines when the revision safety jobs run.	-	Cron Expression
at.risedev.ArchiveKeeper.rekey.maxRetries	How often a rekey binary task (encrypting a binary with another key) is tried if it failed.	3	int
at.risedev.ArchiveKeeper.sync	The configuration for syncing the search connectors with the operation database.		
cron	The cron-expression defines when the synch jobs run.	-	Cron Expression
fetchSize	How many document ids to synch are fetched in one database request.	50	int
batches	How many batches of documents are synched in one run of the synch job. The number of document synched in one rune is the number of batches x fetchSize.	10	int
at.risedev.ArchiveKeeper.connectors.search.sql	The configuration of the SQL search connector		
enabled	If the sql search connector should be used or not	true	boolean
mode	The mode in which the connector is run. READ_ONLY means that no new documents are synched but existing documents can be found. WRITE_ONLY means that no new documents are synched but it is not used in the search. The ONLY modes are helpful when switching the search connector and the data needs to be synced over some time.	READ_WRITE	READ_WRITE, READ_ONLY, WRITE_ONLY
at.risedev.ArchiveKeeper.connectors.kms.mode	Defining which KMS connector is used. Full completely encrypted the data in the remote services. key-only modes encrypt a key in the remote services and use this key for the encryption of the data in Archiekeeper.		remote-key-only, remote-full, aws-key-only
at.risedev.ArchiveKeeper.cryptography			
iv	The initialization vector used in the encryption. Used with key-only mode.	0123456789ABCDEF	
Cipher	The algorithm used for encrypting the data. Used with key-only mode.	AES/CFB/NoPadding	

at.risedev.ArchiveKeeper.connectors.kms.remote	The configuration how to interract with remote KMS		
url	The URL under which the remote KMS can be found. Without a set value the startup will fail if kms.mode is one of the remote mods.		
username	The username used to authenticate the ArchiveKeeper to the KMS. If no username or password is set an unauthenticated request is sent.		
password	The password used to authenticate the ArchiveKeeper to the KMS. If no username or password is set an unauthenticated request is sent.		
at.risedev.ArchiveKeeper.connectors.kms.aws.kmsKeyId	Only used with kms.mode aws-key-only.	alias/arkdev01-parent	
server.error.includeStacktrace	If the value is never the REST error responses will not include a stacktrace and for responses with 500 >= status <= 599 the message and error field will be replaced by a generic value. This way no internal information is exposed to third parties.	NEVER	NEVER, ALWAYS, ON_PARAM
spring.datasource.tomcat	Configuration of database connection pool		
minIdle	The minimum number of connections that should be kept in the pool at all times.	10	
maxIdle	The maximum number of connections that should be kept in the pool at all times.	20	
maxActive	The maximum number of active connections.	20	
initialSize	The initial number of connections.	10	
jdbcInterceptors	A semicolon separated list of classnames extending org.apache.tomcat.jdbc.pool.JdbcInterceptor class. You propaply only want to adjust the slow query threshold which values is in milliseconds.	SlowQueryReport(threshold=500);ResetAbandonedTimer	JDBC Interceptors
dbScheduler	The configuration of the scheduler handling the jobs and tasks e.g. the pruning job.		
enabled	If the scheduler is active in other words if jobs and tasks are executed.	true	boolean

threads	How many threads the scheuler can start to execute the scheduled jobs/tasks.	3	
shutdownMaxWait	The maximum time that a job or task has to gracefuly stop on a shutdown signal. If it takes longer the thread is killed.	5m	
failureLoggerLevel	On which level failures should be logged.	ERROR	
at.risedev.ArchiveKeeper.openapiSpec	The configuration of the openapi REST-API documentation		
Enabled	If the OpenAPI documentation is exposed.	true	boolean
allowUnauthenticated	If unauthenticated callers can retrieve the REST-API documentation.	false	boolean

Note the property files use camel case and the property names in the table are in camel case. If you want to use yaml files transform the names in kebab-case.

Available profiles

Profile name	Description
Web	A pod with this profile is only there for handling user requests. It deactivates the db-scheduler and will not run any scheduled jobs. This means it doesn't execute the periodical jobs. Reduces the default database connection pool size. Best is used together with another pod that uses the scheduler profile.
Scheduler	A pod with this profile is only there for executing the scheduled (periodical) jobs. Reduces the default database connection pool size. Best is used together with another pod that uses the web profile.

COMPRISE GmbH
Wiedner Hauptstraße 76/2
1040 Wient
Austria, Europe

https://www.comprise.world
info@comprise.world