nextcloudappstore Documentation

Release 2.0.0

Bernhard Posselt, Adi Sieker

Dec 26, 2018

Contents

Nextcloud's App Store is an Open Source implementation for hosting Nextcloud apps and their release information with a high focus on developer convenience, stability and usability.

The basic ideas that form the foundation of the store include:

- Be free and open: The App Store is available under the AGPLv3 or later which offers a strong copyleft so developers can profit from changes made in other versions
- Be easy to use: The App Store should be easy to use and be built in a way that users can quickly discover the most loved apps. Registration should be as easy as possible and connect users and developers by using e.g. GitHub or BitBucket logins
- Be DRY (Do not Repeat Yourself): App information should be parsed from the release archive rather than requiring developers to re-enter all the information over and over again
- Validate early: An app release should not be published without validating it beforehand. Package structure, checksums and app metadata can contain mistakes which are often discovered by users rather than developers
- Guide users: Comments sections are often used to post bug reports. This creates additional work for the developer. Therefore appropriate measures should be taken to redirect users to the correct places so issues are resolved faster and in a more convenient way
- Be hard to abuse: Rating abuse should be hard. Therefore rating systems must not be overly complex so counter measures can be taken easily
- Be well documented: APIs and developer use-cases should be documented in such a way that developers and users alike can easily discover the things they need

CHAPTER 1

App Developer Documentation

Look here if you want to upload your own apps or use the REST API

1.1 App Developer Guide

Most of today's developers publish their source code on GitHub, BitBucket, GitLab or on their own GitLab instance. These tools typically also provide a way to release new versions based on Git tags or by uploading custom archives.

Experienced users and package maintainers typically prefer to download the app directly from these services whereas administrators or novice users look for app releases on the App Store. This means that you have to take care of publishing two releases on two different platforms.

We want to avoid duplication and make it harder to ship broken releases by mistake, therefore we went for the following solution:

- Your app's source code is hosted on GitHub or a similar service
- You should use Git tags to create new releases on these services
- GitHub release downloads do not match the required folders structure. This is because GitHub appends a version to the top folder name. Therefore you need to create a separate release which conforms to the expected structure.

This keeps your repository up to date and satisfies the needs of maintainers, developers and experienced users.

1.1.1 Publishing Apps on the App Store

Hosting the archive on a different host means of course that we can not guarantee that the contents have not been tampered with. Neither can we guarantee that the actual app developer uploaded the app. Therefore we require you to sign your app using a certificate.

Obtaining a Certificate

The certificates should be stored in **~/.nextcloud/certificates/** so first create the folder if it does not exist yet:

```
mkdir -p ~/.nextcloud/certificates/
```

Then change into the directory:

```
cd ~/.nextcloud/certificates/
```

and generate your private certificate and CSR:

```
openssl req -nodes -newkey rsa:4096 -keyout APP_ID.key -out APP_ID.csr -subj "/CN=APP_
↪ID"
```

Replace **APP_ID** with your app id, e.g. if your app had an id called **news** you would execute the following command:

```
openssl req -nodes -newkey rsa:4096 -keyout news.key -out news.csr -subj "/CN=news"
```

Note: Keep in mind that an app id must only contain lowercase ASCII characters and underscores!

Then post the contents of your **APP_ID.csr** (e.g. **~/.nextcloud/certificates/news.csr**) on on our certificate repository as pull request and configure your GitHub account to show your mail address in your profile.

We might ask you for further information to verify that you're the legitimate owner of the application. Make sure to keep the private key file (**APP_ID.key**, e.g. **~/.nextcloud/certificates/news.key**) secret and not disclose it to any third-parties.

After we approved your certificate, we will post your signed public certificate (APP_ID.crt) as a response in your app's directory. Take the contents and store it in the same folder with the file name **APP_ID.crt** (e.g. **~/.nextcloud/certificates/news.crt**). Make sure to get rid of excess whitespace at the beginning and end of your file. Your public signed certificate's file contents should look similar to this:

```
-----BEGIN CERTIFICATE-----
MIID+TCCAeECAhAMMA0GCSqGSIb3DQEBCwUAMG0xCzAJBgNVBAYTAlVTMQ8wDQYD
VQQIDAZCb3N0b24xFjAUBgNVBAoMDW93bkNsb3VkIEluYy4xNTAzBgNVBAMMLG93
bkNsb3VkIENvZGUgU2lnbmluZyBJbnRlcm1lZGlhdGUgQXV0aG9yaXR5MB4XDTE2
MDcyNjEwMTIyOFoXDTI2MDcyNDEwMTIyOFowFzEVMBMGA1UEAwwMZm9sZGVycGxh
eWVyMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA8BnaiY7+oPMmYalU
Cpv/U+36PUTQd3r9t73l7opUyv7F2yshrgKk9jdINOWZaPYxFi5mSnolu6KP/nNq
Bsh7HTHFo9xmVg2lia4WxmO23GBp94GEj4irYSP3FcrrT+aLBmr3sM2zxfIWJ9K/
9wC8rFhyQjMaQLqC48VRjz8eI6rRSAUrcY+B6GAB0O2XZifSYVgzwh3lV1Xno2uT
69+V5HfXEEz8u5YRnoFBC8hfaRzGlnm0cUZrVEgEcCjt1pPf+HeGUnHafT8uUbET
7Ys6QCQoaiKy7D7eiUh2kOOcChFAxiGX+9ahiIESZUrlDs8m8rmoa8C3fqho4C9g
nwEoowIDAQABMA0GCSqGSIb3DQEBCwUAA4ICAQAZts21nfQGzkPsiDseIZjg1Dh7
KavuEjxJHqSbTlqi1W9CxievQF205IbfuRLsbsi1Kw1hFivse//nTkFiMvgdqKsM
zSzsUq24tjWFDpNVHgVoPCBG6t7yYP6PdWNZtPSt76w8l7fAo9Fm2tBlFvMfF4Pe
3nveZjV5ns7loFpxLJobl25xj4Q63DKRoN0vVv6bEe+rbd6REPI+Ep8w43A8/wqc
pB0q6j3Fs4FRlNUqshLaRN2HVbllb/+hlA1REOBGEvAuSHzXrThCS2PpEY8Ds7IG
0rSuEdzwCd3c+vk+pssgxmFHBDPDJUsKSgUCF5wzA4k42tK/sixSDJlPcVKZdRrY
+8XgaruPdIMoIVZHXdeNvBtra1kYRxZbeCpe1zSOiLL/xjSWVYEFhiO7ZBuHRDrq
gaJmQNZxzwEUrpLsN4QB4S3jVmCEZ9Rjp8hWuaShRBWwYjlfhKlRcdwCol/T7ODC
oioO3wBapwvsaCS4gmkmdBtvIKvbr62PM2nh6QpJwpyv9LPEXM5ZV0BT3AK8DIK6
ThH5+uRF0QgDXHWIR55Gmh2usJ6VluPWT+f81Q3lH/jxXJfagGOFEFHtyT0yo23M
iazev6j9O2En+uDYLSWgQ7uN+cFYSdfjj1FRjsQ84e8CwNJ1nhiQ/HexMN8zwqDo
6LboOQuGiCet+KggAg==
-----END CERTIFICATE-----
```

Note: Be sure to follow the directory and naming structure for certificates. All our documentation examples and tools will assert this structure.

Registering an App

After you've obtained your signed public certificate you can use it to register your app id on the App Store. To do that either use the *REST API* or use the App Store's register app web interface.

The interface will ask you for the following things:

- **Certificate**: Paste in the contents of your public certificate, e.g. **~/.nextcloud/certificates/news.crt**
- **Signature**: A signature over your app id to verify that you own the private certificate. Can be calculated by using the following command:

  ```
  echo -n "APP_ID" | openssl dgst -sha512 -sign ~/.nextcloud/certificates/APP_ID.
  ↪key | openssl base64
  ```

  where **APP_ID** is your app's id, e.g:

  ```
  echo -n "news" | openssl dgst -sha512 -sign ~/.nextcloud/certificates/news.key |␣
  ↪openssl base64
  ```

We will then verify the certificate and signature and register you as the app's owner. You are now able to publish releases.

Uploading an App Release

After you've registered your app you can upload your app's releases to the App Store. To do that either use the *REST API* or use the App Store's upload app release web interface.

The interface will ask you for the following things:

- **Download**: A download link to your app release archive (tar.gz)
- **Nightly**: Check if you are uploading a nightly release
- **Signature**: A signature over your release archive. Can be calculated by using the following command:

  ```
  openssl dgst -sha512 -sign ~/.nextcloud/certificates/APP_ID.key /path/to/app.tar.
  ↪gz | openssl base64
  ```

  where **APP_ID** is your app's id, e.g:

  ```
  openssl dgst -sha512 -sign ~/.nextcloud/certificates/news.key /path/to/news.tar.
  ↪gz | openssl base64
  ```

We then download the archive and verify the signature. In addition we try to verify and use as much information as possible form the archive, e.g.:

- The archive must only contain one top level folder consisting of lower case ASCII characters and underscores
- The archive must contain an **info.xml** file inside the **appinfo** directory which in turn is located in the top folder
- The info.xml is reformatted using XSLT to bring everything into the correct order (required for XSD 1.0) and unknown elements are dropped. Old elements are migrated to their new equivalents if possible. Afterwards we validate it using an XML Schema (see *Schema Integration*)

If everything went well the release is then either created or updated. The downloaded archive will be deleted from our server.

Updating and Revoking a Certificate

If you've lost or leaked your private certificate you want to revoke and update your certificate:

- Post a new CSR for an already existing app on our issue tracker (edit the same file)
- We will revoke your old certificate and sign your new certificate request
- Then re-register your app certificate on the app register page. This will delete all existing releases.

After you've obtained a new certificate, simply use it to register your app id again (only owners are allowed to do this). This will delete all previous releases from our server since their signature has become invalid.

Transferring Your App to a New Owner

Transferring an app works similar to *registering an app*: The new owner simply needs to register the app again using the public certificate and the signature.

However by default this is restricted to the app's owner. To disable this restriction you first need to unlock your app for the owner transfer. You can do this by going to your **account** settings and choosing Transfer app ownership. On that page you can lock or unlock your apps for being transferred.

After you unlocked your app for transfer, the new owner can then proceed to register the app again. If everything went fine the app is now transferred to the new owner and the transfer setting for that app is locked again.

1.1.2 App Metadata

App metadata is currently being read from the **appinfo/info.xml** and **CHANGELOG.md** file.

info.xml

The info.xml is validated using an XML Schema which can be accessed online.

A minimum valid **info.xml** would look like this:

```xml
<?xml version="1.0"?>
<info xmlns:xsi= "http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation="https://apps.nextcloud.com/schema/apps/info.xsd">
    <id>news</id>
    <name>News</name>
    <summary>An RSS/Atom feed reader</summary>
    <description>An RSS/Atom feed reader</description>
    <version>8.8.2</version>
    <licence>agpl</licence>
    <author>Bernhard Posselt</author>
    <category>multimedia</category>
    <bugs>https://github.com/nextcloud/news/issues</bugs>
    <dependencies>
        <nextcloud min-version="10"/>
    </dependencies>
</info>
```

A full blown example would look like this (needs to be utf-8 encoded):

```
<?xml version="1.0"?>
<info xmlns:xsi= "http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation="https://apps.nextcloud.com/schema/apps/info.xsd">
    <id>news</id>
    <name lang="de">Nachrichten</name>
    <name>News</name>
    <summary lang="en">An RSS/Atom feed reader</summary>
    <description lang="en"># Description\nAn RSS/Atom feed reader</description>
    <description lang="de"><![CDATA[# Beschreibung\nEine Nachrichten App, welche mit␣
↪[RSS/Atom](https://en.wikipedia.org/wiki/RSS) umgehen kann]]></description>
    <version>8.8.2</version>
    <licence>agpl</licence>
    <author mail="mail@provider.com" homepage="http://example.com">Bernhard Posselt</
↪author>
    <author>Alessandro Cosentino</author>
    <author>Jan-Christoph Borchardt</author>
    <documentation>
        <user>https://github.com/nextcloud/news/wiki#user-documentation</user>
        <admin>https://github.com/nextcloud/news#readme</admin>
        <developer>https://github.com/nextcloud/news/wiki#developer-documentation</
↪developer>
    </documentation>
    <category>multimedia</category>
    <category>tools</category>
    <website>https://github.com/nextcloud/news</website>
    <discussion>https://your.forum.com</discussion>
    <bugs>https://github.com/nextcloud/news/issues</bugs>
    <repository>https://github.com/nextcloud/news</repository>
    <screenshot small-thumbnail="https://example.com/1-small.png">https://example.com/
↪1.png</screenshot>
    <screenshot>https://example.com/2.jpg</screenshot>
    <dependencies>
        <php min-version="5.6" min-int-size="64"/>
        <database min-version="9.4">pgsql</database>
        <database>sqlite</database>
        <database min-version="5.5">mysql</database>
        <command>grep</command>
        <command>ls</command>
        <lib min-version="2.7.8">libxml</lib>
        <lib>curl</lib>
        <lib>SimpleXML</lib>
        <lib>iconv</lib>
        <nextcloud min-version="9" max-version="10"/>
    </dependencies>
    <background-jobs>
        <job>OCA\DAV\CardDAV\Sync\SyncJob</job>
    </background-jobs>
    <repair-steps>
        <pre-migration>
            <step>OCA\DAV\Migration\Classification</step>
        </pre-migration>
        <post-migration>
            <step>OCA\DAV\Migration\Classification</step>
        </post-migration>
        <live-migration>
            <step>OCA\DAV\Migration\GenerateBirthdays</step>
        </live-migration>
```

(continues on next page)

(continued from previous page)

```
        <install>
            <step>OCA\DAV\Migration\GenerateBirthdays</step>
        </install>
        <uninstall>
            <step>OCA\DAV\Migration\GenerateBirthdays</step>
        </uninstall>
    </repair-steps>
    <two-factor-providers>
        <provider>OCA\AuthF\TwoFactor\Provider</provider>
    </two-factor-providers>
    <commands>
        <command>A\Php\Class</command>
    </commands>
    <settings>
        <admin>OCA\Theming\Settings\Admin</admin>
        <admin-section>OCA\Theming\Settings\Section</admin-section>
        <personal>OCA\Theming\Settings\Personal</personal>
        <personal-section>OCA\Theming\Settings\PersonalSection</personal-section>
    </settings>
    <activity>
        <settings>
            <setting>OCA\Files\Activity\Settings\FavoriteAction</setting>
            <setting>OCA\Files\Activity\Settings\FileChanged</setting>
            <setting>OCA\Files\Activity\Settings\FileCreated</setting>
            <setting>OCA\Files\Activity\Settings\FileDeleted</setting>
            <setting>OCA\Files\Activity\Settings\FileFavorite</setting>
            <setting>OCA\Files\Activity\Settings\FileRestored</setting>
        </settings>

        <filters>
            <filter>OCA\Files\Activity\Filter\FileChanges</filter>
            <filter>OCA\Files\Activity\Filter\Favorites</filter>
        </filters>

        <providers>
            <provider>OCA\Files\Activity\FavoriteProvider</provider>
            <provider>OCA\Files\Activity\Provider</provider>
        </providers>
    </activity>
    <navigations>
        <navigation role="admin">
            <id>files</id>
            <name>Files</name>
            <route>files.view.index</route>
            <order>0</order>
            <icon>app.svg</icon>
            <type>link</type>
        </navigation>
    </navigations>
    <collaboration>
        <plugins>
            <plugin type="collaborator-search" share-type="SHARE_TYPE_CIRCLE">
↪OCA\Circles\Collaboration\v1\CollaboratorSearchPlugin</plugin>
            <plugin type="autocomplete-sort">OCA\Circles\Collaboration\v1\CircleSorter
↪</plugin>
        </plugins>
    </collaboration>
```

(continues on next page)

(continued from previous page)

```
    <sabre>
        <collections>
            <collection>\OCA\Deck\Dav\RootCollection</collection>
        </collections>
        <plugins>
            <plugin>\OCA\Deck\Dav\ServerPlugin</plugin>
        </plugins>
    </sabre>
</info>
```

The following tags are validated and used in the following way:

id

- required
- must contain only lowercase ASCII characters and underscore
- must match the first folder in the archive
- will be used to identify the app

name

- required
- must occur at least once with **lang="en"** or no lang attribute
- can be translated by using multiple elements with different **lang** attribute values, language code needs to be set **lang** attribute
- will be rendered on the app detail page

summary

- optional
- if not provided the description element's text will be used
- must occur at least once with **lang="en"** or no lang attribute
- can be translated by using multiple elements with different **lang** attribute values, language code needs to be set **lang** attribute
- will be rendered on the app list page as short description

description

- required
- must occur at least once with **lang="en"** or no lang attribute
- can contain Markdown
- can be translated by using multiple elements with different **lang** attribute values, language code needs to be set **lang** attribute
- will be rendered on the app detail page

version

- required
- must be a semantic version without build metadata, e.g. 9.0.1 or 9.1.0-alpha.1

licence

- required
- must contain **agpl**, **mpl*** and/or **apache** as the only valid values. These refer to the AGPLv3, MPL 2.0 and Apache License 2.0

author

- required
- can occur multiple times with different authors
- can contain a **mail** attribute which must be an email
- can contain a **homepage** which must be an URL
- will not (yet) be rendered on the App Store
- will be provided through the REST API

documentation/user

- optional
- must contain an URL to the user documentation
- will be rendered on the app detail page

documentation/admin

- optional
- must contain an URL to the admin documentation
- will be rendered on the app detail page

documentation/developer

- optional
- must contain an URL to the developer documentation
- will be rendered on the app detail page

category

- optional
- if not provided the category **tools** will be used
- must contain one of the following values:
  - **customization**
  - **files**
  - **games**
  - **search**
  - **integration**
  - **monitoring**
  - **multimedia**
  - **office**
  - **organization**
  - **security**

    - **social**
    - **tools**
- old categories are migrated:
    - **auth** will be converted to **security**
- can occur more than once with different categories

website

- optional
- must contain an URL to the project's homepage
- will be rendered on the app detail page

discussion

- optional
- must contain an URL to the project's discussion page/forum
- will be rendered on the app detail page as the "ask question or discuss" button
- if absent, it will default to our forum at https://help.nextcloud.com/ and create a new category in the apps category

bugs

- required
- must contain an URL to the project's bug tracker
- will be rendered on the app detail page

repository

- optional
- must contain an URL to the project's repository
- can contain a **type** attribute, **git**, **mercurial**, **subversion** and **bzr** are allowed values, defaults to **git**
- currently not used

screenshot

- optional
- must contain an HTTPS URL to an image
- can contain a **small-thumbnail** attribute which must contain an https url to an image. This image will be used as small preview (e.g. on the app list overview). Keep it small so it renders fast
- will be rendered on the app list and detail page in the given order

dependencies/php

- optional
- can contain a **min-version** attribute (maximum 3 digits separated by dots)
- can contain a **max-version** attribute (maximum 3 digits separated by dots)
- can contain a **min-int-size** attribute, 32 or 64 are allowed as valid values
- will be rendered on the app releases page

dependencies/database

- optional
- must contain the database name as text, **sqlite**, **pgsql** and **mysql** are allowed as valid values
- can occur multiple times with different databases
- can contain a **min-version** attribute (maximum 3 digits separated by dots)
- can contain a **max-version** attribute (maximum 3 digits separated by dots)
- will be rendered on the app releases page

dependencies/command

- optional
- must contain a linux command as text value
- can occur multiple times with different commands
- will be rendered on the app releases page

dependencies/lib

- optional
- will be rendered on the app releases page
- must contain a required php extension
- can occur multiple times with different php extensions
- can contain a **min-version** attribute (maximum 3 digits separated by dots)
- can contain a **max-version** attribute (maximum 3 digits separated by dots)

dependencies/nextcloud

- required on Nextcloud 11 or higher
- if absent white-listed owncloud versions will be taken from the owncloud element (see below)
- must contain a **min-version** attribute (maximum 3 digits separated by dots)
- can contain a **max-version** attribute (maximum 3 digits separated by dots)

background-jobs/job

- optional
- must contain a php class which is run as background jobs
- will not be used, only validated

repair-steps/pre-migration/step

- optional
- must contain a php class which is run before executing database migrations
- will not be used, only validated

repair-steps/post-migration/step

- optional
- must contain a php class which is run after executing database migrations
- will not be used, only validated

repair-steps/live-migration/step

- optional
- must contain a php class which is run after executing post-migration jobs
- will not be used, only validated

repair-steps/install/step

- optional
- must contain a php class which is run after installing the app
- will not be used, only validated

repair-steps/uninstall/step

- optional
- must contain a php class which is run after uninstalling the app
- will not be used, only validated

two-factor-providers/provider

- optional
- must contain a php class which is registered as two factor auth provider
- will not be used, only validated

commands/command

- optional
- must contain a php class which is registered as occ command
- will not be used, only validated

activity/settings/setting

- optional
- must contain a php class which implements OCPActivityISetting and is used to add additional settings ui elements to the activity app

activity/filters/filter

- optional
- must contain a php class which implements OCPActivityIFilter and is used to add additional filters to the activity app

activity/providers/provider

- optional
- must contain a php class which implements OCPActivityIProvider and is used to react to events from the activity app

settings/admin

- optional
- must contain a php class which implements OCPSettingsISettings and returns the form to render for the global settings area

settings/admin-section

- optional

- must contain a php class which implements OCPSettingsISection and returns data to render navigation entries in the global settings area

settings/personal

- optional
- must contain a php class which implements OCPSettingsISettings and returns the form to render for the global settings area

settings/personal-section

- optional
- must contain a php class which implements OCPSettingsISection and returns data to render navigation entries in the global settings area

navigations

- optional
- must contain at least one navigation element

navigations/navigation

- required
- must contain a name and route element
- denotes a navigation entry
- role denotes the visibility, all means everyone can see it, admin means only an admin can see the navigation entry, defaults to all

navigations/navigation/id

- optional
- the app id
- you can also create entries for other apps by setting an id other than your app one's

navigations/navigation/name

- required
- will be displayed below the navigation entry icon
- will be translated by the default translation tools

navigations/navigation/route

- required
- name of the route that will be used to generate the link

navigations/navigation/icon

- optional
- name of the icon which is looked up in the app's **img/** folder
- defaults to app.svg

navigations/navigation/order

- optional
- used to sort the navigation entries

- a higher order number means that the entry will be ordered further to the bottom

navigations/navigation/type

- optional
- can be either link or settings
- link means that the entry is added to the default app menu
- settings means that the entry is added to the right-side menu which also contains the personal, admin, users, help and logout entry

collaboration

- optional
- can contain plugins for collaboration search (e.g. supplying share dialog)

collaboration/plugins

- optional
- must contain at least one plugin

collaboration/plugins/plugin

- required
- the PHP class name of the plugin
- **must contain type attribute which can be**
  - *collaboration-search* (The class must implement OCPCollaborationCollaboratorsISearchPlugin), requires **share-type** attribute
  - *autocomplete-sort* (The class must implement OCPCollaborationAutoCompleteISorter)
- optionally contain **share-type** attribute

sabre

- optional
- can contain plugins or collections to be loaded by the dav app

sabre/plugins

- optional
- must contain at least one plugin
- A sabre plugin extend the dav system by adding additional event handlers. For mor details see http://sabre.io/dav/writing-plugins/

sabre/plugins/plugin

- required
- the PHP class name of the plugin

sabre/collections

- optional
- must contain at least one collection
- Collections allow apps to expose their own directory tree to the dav endpoint. They will be added to the root of the Nextcloud dav tree.

sabre/collections/collection

- required
- the PHP class name of the plugin
- classes must implement the SabreDAVICollection interface

The following character maximum lengths are enforced:

- All description Strings are database text fields and therefore not limited in size
- All other Strings have a maximum of 256 characters

The following elements are either deprecated or for internal use only and will fail the validation if present:

- **standalone**
- **default_enable**
- **shipped**
- **public**
- **remote**
- **requiremin**
- **requiremax**

database.xml

The database.xml is validated using an XML Schema which can be accessed through the App Store.

A minimum valid **database.xml** would look like this:

```
<?xml version="1.0"?>
<database xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:noNamespaceSchemaLocation="https://apps.nextcloud.com/schema/apps/
↪database.xsd">
    <table>
        <name>*dbprefix*blog_articles</name>
        <declaration>

        </declaration>
    </table>
</database>
```

A full blown example would look like this (needs to be utf-8 encoded):

```
<?xml version="1.0"?>
<database xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:noNamespaceSchemaLocation="https://apps.nextcloud.com/schema/apps/
↪database.xsd">
    <table>
        <name>*dbprefix*blog_articles</name>
        <declaration>
            <field>
                <name>id</name>
                <type>integer</type>
                <length>8</length>
                <unsigned>true</unsigned>
```

(continues on next page)

(continued from previous page)

```
                <notnull>true</notnull>
                <autoincrement>true</autoincrement>
            </field>
            <field>
                <name>user</name>
                <type>text</type>
                <length>255</length>
                <notnull>true</notnull>
                <default>anonymous</default>
            </field>
            <field>
                <name>donations_in_euros</name>
                <type>decimal</type>
                <default>0.00</default>
                <precision>12</precision>
                <scale>2</scale>
            </field>
            <index>
                <name>blog_articles_id_user_index</name>
                <primary>true</primary>
                <unique>true</unique>
                <field>
                    <name>id</name>
                </field>
                <field>
                    <name>user</name>
                </field>
            </index>
            <index>
                <name>blog_articles_user_index</name>
                <field>
                    <name>user</name>
                </field>
            </index>
        </declaration>
    </table>
</database>
```

Note: While you might encounter valid elements like **create**, **overwrite**, **charset** or **sorting** they are not parsed by Nextcloud and can therefore be omitted safely

Changelog

The changelog has to be named **CHANGELOG.md** and being placed in your app's top level folder, e.g. **news/CHANGELOG.md**.

Changelogs have to follow the Keep a CHANGELOG format, e.g.:

```
## [Unreleased]
### Added
- Nighly changes here

## 0.6.0 - 2016-09-20
```

(continues on next page)

(continued from previous page)

```
### Added
- Alias support
  [#1523](https://github.com/owncloud/mail/pull/1523) @tahaalibra
- New incoming messages are prefetched
  [#1631](https://github.com/owncloud/mail/pull/1631) @ChristophWurst
- Custom app folder support
  [#1627](https://github.com/owncloud/mail/pull/1627) @juliushaertl
- Improved search
  [#1609](https://github.com/owncloud/mail/pull/1609) @ChristophWurst
- Scroll to refresh
  [#1595](https://github.com/owncloud/mail/pull/1593) @ChristophWurst
- Shortcuts to star and mark messages as unread
  [#1590](https://github.com/owncloud/mail/pull/1590) @ChristophWurst
- Shortcuts to select previous/next messsage
  [#1557](https://github.com/owncloud/mail/pull/1557) @ChristophWurst

### Changed
- Minimum server is Nextcloud 10/ownCloud 9.1
  [#84](https://github.com/nextcloud/mail/pull/84) @ChristophWurst
- Use session storage instead of local storage for client-side cache
  [#1612](https://github.com/owncloud/mail/pull/1612) @ChristophWurst
- When deleting the current message, the next one is selected immediatelly
  [#1585](https://github.com/owncloud/mail/pull/1585) @ChristophWurst

### Fixed
- Client error while composing a new message
  [#1609](https://github.com/owncloud/mail/pull/1609) @ChristophWurst
- Delay app start until page has finished loading
  [#1634](https://github.com/owncloud/mail/pull/1634) @ChristophWurst
- Auto-redirection of HTML mail links
  [#1603](https://github.com/owncloud/mail/pull/1603) @ChristophWurst
- Update folder counters when reading/deleting messages
  [#1585](https://github.com/owncloud/mail/pull/1585)

### Removed
- Removed old API

### Deprecated
- Deprecated new API

### Security
- Fixed XXE in xml upload
```

Note: The regex for matching the line is **^## (\d+.\d+.\d+)**, the regex for nightlies is **^## [Unreleased]**

The version has to be equal to the version in your info.xml. If the parser can't find a changelog entry, it will be set to an empty string. Only the changelog for the current release will be imported.

The changelog for nightlies will be taken from the **## [Unreleased]** block

Changelogs can be translated as well. To add a changelog for a specific translation, use **CHANGELOG.code.md**, e.g.: **CHANGELOG.fr.md**

1.1.3 Blacklisted Files

To prevent you from nuking your local app's version control directory all uploaded archives are validated to not contain the following folders:

- **.git**

1.1.4 Schema Integration

We provide an XML schema which can be used to validate and get IDE autocompletion for the following files:

- **appinfo/info.xml**:

```
<?xml version="1.0"?>
<info xmlns:xsi= "http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation="https://apps.nextcloud.com/schema/
↪apps/info.xsd">

      <!-- content here -->

</info>
```

- **appinfo/database.xml**:

```
<?xml version="1.0"?>
<database xmlns:xsi= "http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation="https://apps.nextcloud.com/schema/
↪apps/database.xsd">

      <!-- content here -->

</database>
```

You can also validate your info.xml using various online tools

1.2 Store REST API

A REST API for publishing and deleting app releases has been built into the store from day one to help release automation.

1.2.1 API Stability Contract

The API level **will change** if the following occurs:

- a required HTTP request header is added
- a required request parameter is added
- a JSON field of a response object is removed
- a JSON field of a response object is changed to appear optionally
- a JSON field of a response object is changed to a different datatype
- an explicitly documented HTTP response header is removed
- an explicitly documented HTTP response header is changed to a different datatype

- the meaning of an API call changes

The API level **will not change** if:

- a new HTTP response header is added
- an optional new HTTP request header is added
- a new response parameter is added
- the order of the JSON attributes is changed
- if app validation after uploading an app release is changed in any way

You have to design your app with these things in mind:

- Don't depend on the order of object attributes. In JSON it does not matter where the object attribute is since you access the value by name, not by index
- Don't limit your app to the currently available attributes. New ones might be added. If you don't handle them, ignore them
- Use a library to compare versions, ideally one that uses semantic versioning

1.2.2 Authentication

Several routes require authentication. The following authentication methods are supported:

- **Basic**: Http header where **CREDENTIALS** is `base64encode('user:password')`:

  ```
  Authorization: Basic CREDENTIALS
  ```

- **Token**: Http header where **TOKEN** is a token which can be looked up in your account settings or *acquired through the API*:

  ```
  Authorization: Token TOKEN
  ```

Note: If you created your account using GitHub you will always need to use token authentication since we do not have access to your password. The token can be looked up in your account settings

1.2.3 Specification

The following API routes are present:

- *Get API Token*
- *Regenerate API Token*
- *Get All Categories*
- *Get All Nextcloud Releases*
- *Get All Apps and Releases*
- *Register a New App*
- *Publish a New App Release*
- *Delete an App Release*
- *Delete a Nightly App Release*

Get API Token

This route will return the API token for the authenticated user. If no token exists, one will be generated.

- **Url**: POST /api/v1/token
- **Authentication**: Basic, Session
- **Example CURL request**:

```
curl -X POST https://apps.nextcloud.com/api/v1/token -u "user:password"
```

- **Returns**: application/json

```
{"token":"4b92477ff8d5fe889be75db4c7d9a09116276920"}
```

Regenerate API Token

This route will generate and return a new API token for the authenticated user regardless of whether a token already exists.

- **Url**: POST /api/v1/token/new
- **Authentication**: Basic, Token
- **Example CURL request**:

```
curl -X POST https://apps.nextcloud.com/api/v1/token/new -u "user:password"
```

- **Returns**: application/json

```
{"token":"ca3fb97920705d2c2ecdb0900f8ed5cf5744704d"}
```

Get All Categories

This route will return all categories and their translations.

- **Url**: GET /api/v1/categories.json
- **Authentication**: None
- **Caching**: ETag
- **Example CURL request**:

```
curl https://apps.nextcloud.com/api/v1/categories.json -H 'If-None-Match: "4-2016-
↪06-11 10:37:24+00:00"'
```

- **Returns**: application/json

```
[
    {
        "id": "games",
        "translations": {
            "en": {
                "name": "Games",
                "description": ""
            },
            "de": {
                "name": "Spiele",
                "description": ""
            },
            "fr": {
                "name": "Jeux",
                "description": ""
            }
        }
    },
    {
        "id": "multimedia",
        "translations": {
            "en": {
                "name": "Multimedia",
                "description": ""
            },
            "de": {
                "name": "Multimedia",
                "description": ""
            },
            "fr": {
                "name": "Multimedia",
                "description": ""
            }
        }
    },
    {
        "id": "pim",
        "translations": {
            "en": {
                "name": "PIM",
                "description": ""
            },
            "de": {
                "name": "PIM",
                "description": ""
            },
            "fr": {
                "name": "PIM",
                "description": ""
            }
        }
    },
    {
        "id": "tools",
        "translations": {
            "en": {
                "name": "Tools",
```

(continues on next page)

(continued from previous page)

```
                "description": ""
            },
            "de": {
                "name": "Werkzeuge",
                "description": ""
            },
            "fr": {
                "name": "Outil",
                "description": ""
            }
        }
    }
]
```

translations Translated fields are stored inside a translations object. They can have any size, depending on if there is a translation. If a required language is not found, you should fall back to English.

Get All Nextcloud Releases

This will return all the Nextcloud releases that the store knows about. To check if a release can actually be downloaded check the **hasRelease** flag.

Note: Unsupported Nextcloud releases will be removed from the response

Note: To find the latest version that has a release you will need to use a semantic version library to sort the list. The result is unsorted.

- **Url**: GET /api/v1/platforms.json
- **Authentication**: None
- **Caching**: ETag
- **Example CURL request**:

```
curl https://apps.nextcloud.com/api/v1/platforms.json -H 'If-None-Match: "4-2016-
↪06-11 10:37:24+00:00"'
```

- **Returns**: application/json

```
[
    {
        "hasRelease": false,
        "version": "99.0.0",
        "isSupported": true
    },
    {
        "hasRelease": true,
        "version": "9.0.0",
        "isSupported": false
    }
]
```

hasRelease boolean flag that indicates if the Nextcloud release is officially out yet

isSupported boolean flag that indicates if the Nextcloud is officially supported

Get All Apps and Releases

This route will return all releases to display inside Nextcloud's apps admin area.

- **Url**: GET /api/v1/platform/{**platform-version**}/apps.json
- **Url parameters**:
  - **platform-version**: semantic version, digits only: Returns all the apps and their releases that work on this version. If an app has no working releases, the app will be excluded
- **Authentication**: None
- **Caching**: ETag
- **Example CURL request**:

```
curl https://apps.nextcloud.com/api/v1/platform/9.0.0/apps.json -H 'If-None-
↪Match: "1-1-2016-06-17 23:08:58.042321+00:00"'
```

- **Returns**: application/json

```
[
    {
        "id": "news",
        "categories": [
            "multimedia"
        ],
        "authors": [
            {
                "name": "Bernhard Posselt",
                "mail": "",
                "homepage": ""
            },
            {
                "name": "Alessandro Cosentino",
                "mail": "",
                "homepage": ""
            },
            {
                "name": "Jan-Christoph Borchardt",
                "mail": "",
                "homepage": ""
            }
        ],
        "userDocs": "https://github.com/owncloud/news/wiki#user-documentation",
        "adminDocs": "https://github.com/owncloud/news#readme",
        "developerDocs": "https://github.com/owncloud/news/wiki#developer-
↪documentation",
        "issueTracker": "https://github.com/owncloud/news/issues",
        "website": "https://github.com/owncloud/news",
        "discussion": "https://help.nextcloud.com/c/apps/news",
        "created": "2016-06-25T16:08:56.794719Z",
        "lastModified": "2016-06-25T16:49:25.326855Z",
        "ratingOverall": 0.5,
        "ratingNumOverall": 20,
        "ratingRecent": 1.0,
```

(continues on next page)

(continued from previous page)

```
        "ratingNumRecent": 10,
        "releases": [
            {
                "version": "9.0.4-alpha.1",
                "phpExtensions": [
                    {
                        "id": "libxml",
                        "versionSpec": ">=2.7.8",
                        "rawVersionSpec": ">=2.7.8"
                    },
                    {
                        "id": "curl",
                        "versionSpec": "*",
                        "rawVersionSpec": "*"
                    },
                    {
                        "id": "SimpleXML",
                        "versionSpec": "*",
                        "rawVersionSpec": "*"
                    },
                    {
                        "id": "iconv",
                        "versionSpec": "*",
                        "rawVersionSpec": "*"
                    }
                ],
                "databases": [
                    {
                        "id": "pgsql",
                        "versionSpec": ">=9.4.0",
                        "rawVersionSpec": ">=9.4"
                    },
                    {
                        "id": "sqlite",
                        "versionSpec": "*",
                        "rawVersionSpec": "*"
                    },
                    {
                        "id": "mysql",
                        "versionSpec": ">=5.5.0",
                        "rawVersionSpec": ">=5.5"
                    }
                ],
                "shellCommands": [
                    "grep"
                ],
                "phpVersionSpec": ">=5.6.0",
                "platformVersionSpec": ">=9.0.0 <9.2.0",
                "rawPhpVersionSpec": ">=5.6",
                "rawPlatformVersionSpec": ">=10 <=10",
                "minIntSize": 64,
                "isNightly": false,
                "download": "https://github.com/owncloud/news/releases/download/8.8.0/
↪news.tar.gz",
                "created": "2016-06-25T16:08:56.796646Z",
                "licenses": [
                    "agpl"
```

(continues on next page)

(continued from previous page)

```
                ],
                "lastModified": "2016-06-25T16:49:25.319425Z",
                "signature":
↪"909377e1a695bbaa415c10ae087ae1cc48e88066d20a5a7a8beed149e9fad3d5",
                "translations": {
                    "en": {
                        "changelog": "* **Bugfix**: Pad API last modified timestamp␣
↪to milliseconds in updated items API to return only new items. API users however␣
↪need to re-sync their complete contents, #24\n* **Bugfix**: Do not pad milliseconds␣
↪for non millisecond timestamps in API"
                    }
                }
            }
        ],
        "screenshots": [
            {
                "url": "https://example.com/news.jpg",
                "smallThumbnail": ""
            }
        ],
        "translations": {
            "en": {
                "name": "News",
                "summary": "An RSS/Atom feed reader",
                "description": "# This is markdown\nnext line"
            }
        },
        "isFeatured": false,
        "certificate": "-----BEGIN CERTIFICATE-----
↪\r\nMIIEojCCA4qgAwIBAgICEAAwDQYJKoZIhvcNAQELBQAwezELMAkGA1UEBhMCREUx\r\nGzAZBgNVBAgMEkJhZGVuLVd1ZX
↪\r\nBqq1HCmUB6tulnGcxUzt\/Z\/
↪oSIgnuGyENeke077W3EyryINL7EIyD4Xp7sxLizTM\r\nFCFCjjH1AgMBAAGjggFDMIIBPzAJBgNVHRMEAjAAMBEGCWCGSAGG+
↪+0NEH3nahTBFxO6nKyR\/VWigACH0\r\nnaV0ecTcoQwDjKDNNFr+4S1WlHdwITlnNabC7v9rZ\/
↪6QvbkrOTuO9fOR6azp1EwW\r\n2pixWqj0Sb9\/
↪dSIVRpSq+jpBE6JAiX44dSR7zoBxRB8DgVO2Afy0s80xEpr5JAzb\r\nNYuPS7M5UHdAv2dr16fDcDIvn+vk92KpNh1NTeZFjB
↪+QBOgB299QVCKQU+lcZWptQt+RdsJUm46\r\nNY\/
↪nARy4Oi4uOe88SuWITj9KhrFmEvrUlgM8FvoXA1ldrR7KiEg=\r\n-----END CERTIFICATE-----",
        "signatureDigest": "sha512"
    }
]
```

translations Translated fields are stored inside a translations object. They can have any size, depending on if there is a translation. If a required language is not found, you should fall back to English.

isNightly True if the release is a nightly version. New nightly releases are not required to have a higher version than the previous one to be considered greater. Instead look at the **lastModified** attribute to detect updates if both nightly versions are equal. Example: 1.0.0 is equal to 1.0.0, however if the second one has a nightly flag, then the second one is greater. If both versions have nightly flags and are equal, the **lastModified** is used to determine the precedence.

screenshots Guaranteed to be HTTPS

smallThumbnail Small thumbnail which can be used as preview image. Guaranteed to be HTTPS. Not required, so if not present or an empty string, use the screenshot url instead.

download Download archive location, guaranteed to be HTTPS

versionSpec Required versions (minimum and maximum versions) are transformed to semantic version specs. If a field is a *, this means that there is no version requirement. The following permutations can occur:

- **All versions**: *
- **Maximum version only**: <8.1.2
- **Minimum version only**: >=9.3.2
- **Maximum and minimum version**: >=9.3.2 <8.1.2

rawVersionSpec Non semantic versions as they occur in the info.xml. The following permutations can occur:

- **All versions**: *
- **Maximum version only**: <=8.1.2
- **Minimum version only**: >=9.3.2
- **Maximum and minimum version**: >=9.3.2 <=8.1.2

ratingRecent Rating from 0.0 to 1.0 (0.0 being the worst, 1.0 being the best) in the past 90 days

ratingNumRecent Number of ratings for an app in the past 90 days, as in: how many votes were casted. 0 Means no ratings yet.

ratingOverall Rating from 0.0 to 1.0 (0.0 being the worst, 1.0 being the best) of all time

ratingNumOverall Number of ratings for an app overall, as in: how many votes were casted. 0 Means no ratings yet.

signature A signature using SHA512 and the app's certificate

signatureDigest The hashing algorithm that is used to verify the signature

description A full blown description containing Markdown

summary A brief explanation what the app tries to do

isFeatured Simple boolean flag which will be presented to the user as "hey take a look at this app". Does not imply that it has been reviewed or we recommend it officially

categories The string value is the category's id attribute, see *Get All Categories*

changelog The translated release changelog in Markdown. Can be empty for all languages

version A semantic version without build metadata (e.g. 1.3.0, 1.2.1-alpha.1)

Register a New App

Before you can upload release you first need to register its app id. To do that use:

- **Url**: POST /api/v1/apps
- **Authentication** Basic, Token
- **Content-Type**: application/json
- **Request body**:
  - **certificate**: Your public certificate whose CN is equal to the app id, should be stored in **~/.nextcloud/certificates/APP_ID.cert** where **APP_ID** is your app's id
  - **signature**: A SHA512 signature over the app id using the app's certificate, can be created using:

```
echo -n "APP_ID" | openssl dgst -sha512 -sign ~/.nextcloud/certificates/APP_
↪ID.key | openssl base64
```

```
{
    "certificate": "certificate": "-----BEGIN CERTIFICATE-----
↪\r\nMIIEojCCA4qgAwIBAgICEAAwDQYJKoZIhvcNAQELBQAwezELMAkGA1UEBhMCREUx\r\nGzAZBgNVBAgMEkJhZGVuLV
↪\r\nBqq1HCmUB6tulnGcxUzt\/Z\/
↪oSIgnuGyENeke077W3EyryINL7EIyD4Xp7sxLizTM\r\nFCFCjjH1AgMBAAGjggFDMIIBPzAJBgNVHRMEAjAAMBEGCWCGS
↪+0NEH3nahTBFxO6nKyR\/VWigACH0\r\nnaV0ecTcoQwDjKDNNFr+4S1WlHdwITlnNabC7v9rZ\/
↪6QvbkrOTuO9fOR6azp1EwW\r\n2pixWqj0Sb9\/
↪dSIVRpSq+jpBE6JAiX44dSR7zoBxRB8DgVO2Afy0s80xEpr5JAzb\r\nNYuPS7M5UHdAv2dr16fDcDIvn+vk92KpNh1NTe
↪+QBOgB299QVCKQU+lcZWptQt+RdsJUm46\r\nNY\/
↪nARy4Oi4uOe88SuWITj9KhrFmEvrUlgM8FvoXA1ldrR7KiEg=\r\n-----END CERTIFICATE-----",
    "signature": "65e613318107bceb131af5cf8b71e773b79e1a9476506f502c8e2017b52aba15
↪"
}
```

- **Example CURL request**:

```
curl -X POST -u "user:password" https://apps.nextcloud.com/api/v1/apps -H
↪"Content-Type: application/json" -d '{"certificate": "certificate": "-----BEGIN␣
↪CERTIFICATE-----
↪\r\nMIIEojCCA4qgAwIBAgICEAAwDQYJKoZIhvcNAQELBQAwezELMAkGA1UEBhMCREUx\r\nGzAZBgNVBAgMEkJhZGVuLV
↪\r\nBqq1HCmUB6tulnGcxUzt\/Z\/
↪oSIgnuGyENeke077W3EyryINL7EIyD4Xp7sxLizTM\r\nFCFCjjH1AgMBAAGjggFDMIIBPzAJBgNVHRMEAjAAMBEGCWCGS
↪+0NEH3nahTBFxO6nKyR\/VWigACH0\r\nnaV0ecTcoQwDjKDNNFr+4S1WlHdwITlnNabC7v9rZ\/
↪6QvbkrOTuO9fOR6azp1EwW\r\n2pixWqj0Sb9\/
↪dSIVRpSq+jpBE6JAiX44dSR7zoBxRB8DgVO2Afy0s80xEpr5JAzb\r\nNYuPS7M5UHdAv2dr16fDcDIvn+vk92KpNh1NTe
↪+QBOgB299QVCKQU+lcZWptQt+RdsJUm46\r\nNY\/
↪nARy4Oi4uOe88SuWITj9KhrFmEvrUlgM8FvoXA1ldrR7KiEg=\r\n-----END CERTIFICATE-----",
↪"signature": "65e613318107bceb131af5cf8b71e773b79e1a9476506f502c8e2017b52aba15"}
↪'
```

- **Returns**:
  - **HTTP 201**: If the app was not previously present and was registered successfully
  - **HTTP 204**: If the app has been updated (either owner or certificate change)
  - **HTTP 400**: If the app id contains invalid characters, the signature could not be validated or if the posted app certificate has been revoked
  - **HTTP 401**: If the user is not authenticated
  - **HTTP 403**: If the user is not authorized to update the app signature (only owners are allowed to do so)

You can also use this route to register a new certificate for an app if you are the app owner. However keep in mind that this will delete all previous app releases, since their signatures are now invalid and not installable anymore.

Find out more how to generate and request the certificate signature by following the *App Developer Guide*.

Note: **DO NOT** post your private key which is stored in the **.key** file. The private certificate needs to be stored securely. If you are unsure whether a file is a private certificate or the public one: your private certificate's content starts with **—–BEGIN PRIVATE KEY—–**, whereas your public certificate's content starts with **—–BEGIN CERTIFICATE—–**

Note: Keep in mind that we verify that the posted certificate and the signature are valid: the certificate needs to be signed by us and your app id signature must stem from the same certificate and match the expected result.

Publish a New App Release

The following request will create a new app release or update an existing release:

- **Url**: POST /api/v1/apps/releases
- **Authentication** Basic, Token
- **Content-Type**: application/json
- **Request body**:
  - **download**: An Https (Http is not allowed!) link to the archive packaged (maximum size: 20 Megabytes) as tar.gz, info.xml must be smaller than 512Kb
  - **signature**: A SHA512 signature over the archive using the app's certificate, can be created using:

    ```
    openssl dgst -sha512 -sign ~/.nextcloud/certificates/APP_ID.key /path/to/app.
    ↪tar.gz | openssl base64
    ```

  - **nightly (Optional)**: If true this release will be stored as a nightly. All previous nightly releases will be deleted.

  ```
  {
      "download": "https://example.com/release.tar.gz",
      "signature": "65e613318107bceb131af5cf8b71e773b79e1a9476506f502c8e2017b52aba15
  ↪",
      "nightly": false
  }
  ```

- **Example CURL request**:

  ```
  curl -X POST -u "user:password" https://apps.nextcloud.com/api/v1/apps/releases -
  ↪H "Content-Type: application/json" -d '{"download":"https://example.com/release.
  ↪tar.gz", "signature":
  ↪"65e613318107bceb131af5cf8b71e773b79e1a9476506f502c8e2017b52aba15"}'
  ```

- **Returns**:
  - **HTTP 200**: If the app release was updated successfully
  - **HTTP 201**: If the app release was created successfully
  - **HTTP 400**: If the app release contains invalid data, is too large, is not registered yet, the signature could not be validated, the current app certificate has been revoked or could not be downloaded from the provided link
  - **HTTP 401**: If the user is not authenticated
  - **HTTP 403**: If the user is not authorized to create or update the app release

If there is no app with the given app id yet it will fail: you need to *register your app id first*. Then the **info.xml** file which lies in the compressed archive's folder **app-id/appinfo/info.xml** is being parsed and validated. Afterwards the provided signature will be validated using the app's certificate and the downloaded archive's SHA512 checksum. The validated result is then saved in the database. Both owners and co-maintainers are allowed to upload new releases.

If the app release version is the latest version, everything is updated. If it's not the latest release, only release relevant details are updated. This **excludes** the following info.xml elements:

- name
- summary

- description
- category
- author
- documentation
- bugs
- website
- screenshot

For more information about validation and which **info.xml** fields are parsed, see *App Metadata*

Delete an App Release

Only app owners or co-maintainers are allowed to delete an app release. The owner is the user that pushes the first release of an app to the store.

- **Url**: DELETE /api/v1/apps/{**app-id**}/releases/{**app-version**}
- **Url parameters**:
- **app-id**: app id, lower case ASCII characters and underscores are allowed
- **app-version**: app version, semantic version, digits only
- **Authentication**: Basic, Token
- **Authorization**: App owners and co-maintainers
- **Example CURL request**:

```
curl -X DELETE https://apps.nextcloud.com/api/v1/apps/news/releases/9.0.0 -u
↪"user:password"
```

- **Returns**:
  - **HTTP 204**: If the app release was deleted successfully
  - **HTTP 401**: If the user is not authenticated
  - **HTTP 403**: If the user is not authorized to delete the app release
  - **HTTP 404**: If the app release could not be found

Delete a Nightly App Release

Only app owners or co-maintainers are allowed to delete a nightly app release. The owner is the user that pushes the first release of an app to the store.

- **Url**: DELETE /api/v1/apps/{**app-id**}/releases/nightly/{**app-version**}
- **Url parameters**:
- **app-id**: app id, lower case ASCII characters and underscores are allowed
- **app-version**: app version, semantic version, digits only
- **Authentication**: Basic, Token
- **Authorization**: App owners and co-maintainers

- **Example CURL request**:

```
curl -X DELETE https://apps.nextcloud.com/api/v1/apps/news/releases/nightly/9.0.0␣
↪-u "user:password"
```

- **Returns**:
  - **HTTP 204**: If the app release was deleted successfully
  - **HTTP 401**: If the user is not authenticated
  - **HTTP 403**: If the user is not authorized to delete the app release
  - **HTTP 404**: If the app release could not be found

Delete an App

Only app owners are allowed to delete an app. The owner is the user that pushes the first release of an app to the store.

Deleting an app will also delete all releases which are associated with it.

- **Url**: DELETE /api/v1/apps/{**app-id**}
- **Url parameters**:
- **app-id**: app id, lower case ASCII characters and underscores are allowed
- **Authentication**: Basic, Token
- **Authorization**: App owners
- **Example CURL request**:

```
curl -X DELETE https://apps.nextcloud.com/api/v1/apps/news -u "user:password"
```

- **Returns**:
- **HTTP 204**: If the app was deleted successfully
- **HTTP 401**: If the user is not authenticated
- **HTTP 403**: If the user is not authorized to delete the app
- **HTTP 404**: If the app could not be found

Get All App Ratings

This route will return all rating comments.

- **Url**: GET /api/v1/ratings.json
- **Authentication**: None
- **Caching**: ETag
- **Example CURL request**:

```
curl https://apps.nextcloud.com/api/v1/ratings.json -H 'If-None-Match: ""1-2016-
↪09-03 17:11:38.772856+00:00""'
```

- **Returns**: application/json

```
[
    {
        "rating": 1.0,
        "ratedAt": "2016-09-03T17:11:38.772856Z",
        "translations": {
            "en": {
                "comment": "I like it"
            }
        },
        "user": {
            "id": 1,
            "firstName": "Tom",
            "lastName": "Jones"
        },
        "app": "keeweb"
    }
]
```

translations can contain 0 or any number of translated comments. If no comment is available for the currently chosen language, only the rating should be considered. Contains Markdown.

rating range from 0.0 (worst) to 1.0 (best)

CHAPTER 2

App Store Admin Documentation

Look here if you want to install the store on your server and keep it up to date

2.1 Store Production Installation

There are two ways to install the store, both are mutually exclusive (means: don't mix and match). If you are looking for a development setup, proceed to *Store Development Installation*, otherwise continue.

Note: This guide will use Ubuntu 16.04, Apache and PostgreSQL to set up the app store. You can of course also use different distributions and web-servers, however we will not be able to support you.

2.1.1 Installing Packages

First you want to switch your machine to an up to date Node.js version and install Yarn:

```
curl -sS https://deb.nodesource.com/gpgkey/nodesource.gpg.key | sudo apt-key add -
echo "deb https://deb.nodesource.com/node_8.x xenial main" | sudo tee /etc/apt/
↪sources.list.d/nodesource.list
echo "deb-src https://deb.nodesource.com/node_8.x xenial main" | sudo tee -a /etc/apt/
↪sources.list.d/nodesource.list

curl -sS https://dl.yarnpkg.com/debian/pubkey.gpg | sudo apt-key add -
echo "deb https://dl.yarnpkg.com/debian/ stable main" | sudo tee /etc/apt/sources.
↪list.d/yarn.list
```

Then install the following libraries:

```
sudo apt-get update
sudo apt-get install python3-venv python3-wheel libxslt-dev libxml2-dev libz-dev␣
↪libpq-dev build-essential python3-dev python3-setuptools git gettext libssl-dev␣
↪libffi-dev nodejs yarn
```

(continues on next page)

(continued from previous page)

2.1.2 Database Setup

Then install the database:

```
sudo apt-get install postgresql
```

configure it:

```
echo "listen_address = '127.0.0.1'" | sudo tee -a /etc/postgresql/9.5/main/pg_ident.
↪conf
sudo systemctl restart postgresql.service
```

and create a user and database:

```
sudo -s
su - postgres
psql
CREATE USER nextcloudappstore WITH PASSWORD 'password';
CREATE DATABASE nextcloudappstore OWNER nextcloudappstore;
\q
exit
exit
```

Note: Use your own password instead of the password example!

2.1.3 App Store Setup

Before you begin to set up the App Store, make sure that your locales are set up correctly. You can fix your locales by running:

```
export LC_ALL="en_US.UTF-8"
export LC_CTYPE="en_US.UTF-8"
sudo dpkg-reconfigure locales
```

Afterwards change into your preferred target folder, clone the repository using git and change into it:

```
cd /path/to/target
git clone https://github.com/nextcloud/appstore.git
cd appstore
```

Afterwards set up a new virtual environment by running the following command:

```
python3 -m venv venv
```

This will create a local virtual environment in the **venv** folder. You only need to do this once in the beginning.

Then activate it:

```
source venv/bin/activate
```

Note: The above command changes your shell settings for the current session only, so once you launch a new terminal you need to run the command again to register all the paths.

Note: Keep in mind that you need to have the virtual environment activated for all the following commands

2.1.4 Installing Required Libraries

Next install the required libraries:

```
pip install --upgrade wheel
pip install --upgrade pip
pip install -r requirements/base.txt
pip install -r requirements/production.txt
```

2.1.5 Adjusting Default Settings

To get your instance running in production you need to create your production settings file in **nextcloudappstore/settings/production.py** which overwrites and enhances the settings defined in **nextcloudappstore/settings/base.py**. The production settings file is excluded from version control and should contain at least something like the following:

```
from nextcloudappstore.settings.base import *

# DEBUG must be false to not leak sensitive content
DEBUG = False

# generate the SECRET_KEY by yourself for instance by using the following command:
# env LC_CTYPE=C tr -dc "a-zA-Z0-9-_\$\?" < /dev/urandom | head -c 64; echo
SECRET_KEY = 'change this!'

ALLOWED_HOSTS = ['production-domain.com']

# E-Mail settings which are used to send mails (e.g. confirm account messages)
# for more configuration options consult the Django documentation
# https://docs.djangoproject.com/en/1.11/ref/settings/#std:setting-EMAIL_HOST
DEFAULT_FROM_EMAIL = 'admin@yourdomain.com'
ADMINS = [('Your Name', 'your-mail@example.com')]
EMAIL_HOST = 'localhost'

# postgres or other db if needed if anything other than sqlite is used
# you need to create the database, user and password first
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'nextcloudappstore',
        'USER': 'nextcloudappstore',
        'PASSWORD': 'password',
        'HOST': '127.0.0.1',
        'PORT': '5432',
    }
}
```

(continues on next page)

(continued from previous page)

```
# The following lines are HTTPS only!
CSRF_COOKIE_SECURE = True
SESSION_COOKIE_SECURE = True
SECURE_HSTS_SECONDS = 31536000
SECURE_HSTS_INCLUDE_SUBDOMAINS = True
ACCOUNT_DEFAULT_HTTP_PROTOCOL = 'https'
CSP_IMG_SRC = ('https:',)

# Path to where your static content lies (e.g. CSS, JavaScript and images)
# This should point to a directory served by your web-server
STATIC_ROOT = '/var/www/production-domain.com/static/'

# Url for serving content uploaded by users, ideally different domain
MEDIA_URL = 'https://separate-domain.com/'

# Path to where user uploaded content lies, should point to a directory
# served by your web-server
MEDIA_ROOT = '/var/www/production-domain.com/media/'

# Public and private keys for Googles recaptcha
RECAPTCHA_PUBLIC_KEY = 'YOUR PUBLIC KEY'
RECAPTCHA_PRIVATE_KEY = 'YOUR PRIVATE KEY'

LOG_LEVEL = 'ERROR'
LOGGING['handlers']['file']['filename'] = LOG_FILE
LOGGING['handlers']['file']['level'] = LOG_LEVEL
LOGGING['loggers']['django']['level'] = LOG_LEVEL

# Discourse user that is allowed to create categories. This will be used
# to automatically create categories when registering apps
DISCOURSE_USER = 'tom'
DISCOURSE_TOKEN = 'a token'

#########################
# Overridable Defaults: #
#########################

# Url for serving non user uploaded files like CSS, JavaScript and images
# STATIC_URL = '/static/'

# Url or domain for serving user uploaded files
# MEDIA_URL = '/media/'

# how many times a user is allowed to call the app upload route per day
# REST_FRAMEWORK['DEFAULT_THROTTLE_RATES']['app_upload'] = '100/day'
# how many times a user is allowed to call the app register route per day
# REST_FRAMEWORK['DEFAULT_THROTTLE_RATES']['app_register'] = '100/day'

# Only set this parameter if you want to use a different tmp directory for app␣
↪downloads
# RELEASE_DOWNLOAD_ROOT = '/other/tmp'

# Only set if you want a different log location than the one in the main directory
# Make sure that this appears above the first use
# LOG_FILE = '/path/to/appstore/appstore.log'
```

(continues on next page)

(continued from previous page)

```
# minimum number of comments to calculate a rating
# RATING_THRESHOLD = 5

# number of days to include from today in the recent ratings calculation
# RATING_RECENT_DAY_RANGE = 90

# MAX_DOWNLOAD_FILE_SIZE = 1024 ** 2  # bytes
# MAX_DOWNLOAD_TIMEOUT = 60  # seconds
# MAX_DOWNLOAD_REDIRECTS = 10
# MAX_DOWNLOAD_SIZE = 20 * (1024 ** 2)  # bytes
# ARCHIVE_FOLDER_BLACKLIST = {
#     'No .git directories': r'\.git$'
# }

# DISCOURSE_URL = 'https://help.nextcloud.com'

# If given a sub category will be created at this location
# If not given a root category will be created
# You can get the category id here at the /categories.json route, e.g.
# https://help.nextcloud.com/categories.json
# DISCOURSE_PARENT_CATEGORY_ID = 26
```

Then set the file as the active settings file:

```
export DJANGO_SETTINGS_MODULE=nextcloudappstore.settings.production
```

Note: Absolutely make sure to generate a new **SECRET_KEY** value! Use the following command for instance to generate a token:

```
env LC_CTYPE=C tr -dc "a-zA-Z0-9-_\$\?" < /dev/urandom | head -c 64; echo
```

For more settings, check the settings documentation.

2.1.6 Creating the Database Schema

After all settings are adjusted, create the database schema by running the following command:

```
python manage.py migrate
```

2.1.7 Creating an Admin User

To create the initial admin user and verify his email, run the following command:

```
python manage.py createsuperuser --username admin --email admin@admin.com
python manage.py verifyemail --username admin --email admin@admin.com
```

The first command will ask for the password.

2.1.8 Loading Initial Data

To pre-populate the database with categories and other data run the following command:

```
python manage.py loaddata nextcloudappstore/core/fixtures/*.json
```

2.1.9 Initializing Translations

To import all translations run:

```
python manage.py compilemessages
python manage.py importdbtranslations
```

2.1.10 Building the Frontend

To build the frontend run:

```
yarn install
yarn run build
```

2.1.11 Placing Static Content

Django web apps usually ship static content such as JavaScript, CSS and images inside the project folder's apps. In order for them to be served by your web server they need to be gathered and placed inside a folder accessible by your server. To do that first create the appropriate folders:

```
sudo mkdir -p /var/www/production-domain.com/static/
sudo mkdir -p  /var/www/production-domain.com/media/
```

Then copy the files into the folders by executing the following commands:

```
sudo chown -R $(whoami):users /var/www
python manage.py collectstatic
sudo chown -R www-data:www-data /var/www
```

This will place the contents inside the folder configured under the key **STATIC_ROOT** and **MEDIA_ROOT** inside your **nextcloudappstore/settings/production.py**

2.1.12 Configuring the Web-Server

First install Apache and mod_wsgi:

```
sudo apt-get install apache2 libapache2-mod-wsgi-py3
```

Then adjust the config in **/etc/apache2/sites-enabled/default.conf** and add the following configuration to your **VirtualHost** section:

```
<VirtualHost *:80>

WSGIDaemonProcess apps python-home=/path/to/appstore/venv python-path=/path/to/
↪appstore
WSGIProcessGroup apps
WSGIScriptAlias / /path/to/appstore/nextcloudappstore/wsgi.py
WSGIPassAuthorization On
```

(continues on next page)

(continued from previous page)

```
Alias /static/ /var/www/production-domain.com/static/
Alias /schema/apps/info.xsd /path/to/appstore/nextcloudappstore/api/v1/release/info.
↪xsd
Alias /schema/apps/database.xsd /path/to/appstore/nextcloudappstore/api/v1/release/
↪database.xsd

<Directory /path/to/appstore/nextcloudappstore>
    <Files wsgi.py>
        Require all granted
    </Files>
</Directory>

<Directory /path/to/appstore/nextcloudappstore/api/v1/release>
    <Files info.xsd>
        Require all granted
        Header always set X-Content-Type-Options nosniff
        Header always set X-XSS-Protection: 1; mode=block
    </Files>
    <Files database.xsd>
        Require all granted
        Header always set X-Content-Type-Options nosniff
        Header always set X-XSS-Protection: 1; mode=block
    </Files>
</Directory>

<Directory /var/www/production-domain.com/static/>
    Require all granted
    AllowOverride None
    Header always set X-Content-Type-Options nosniff
    Header always set X-XSS-Protection: 1; mode=block
</Directory>

<Directory /var/www/production-domain.com/media/>
    Require all granted
    AllowOverride None
    Header always set X-Content-Type-Options nosniff
    Header always set X-XSS-Protection: 1; mode=block
</Directory>

</VirtualHost>
```

Note: Your configuration will look different depending on where you place your static files and if you enable SSL. This is just a very minimal non HTTPS example.

Note: It could be that you need to enable **mod_headers**. To do this simply run **sudo a2enmod headers**

Finally restart Apache:

```
sudo systemctl restart apache2
```

2.1.13 Logging

Depending on where you have configured the log file location, you need to give your web server access to it. By default the logfile is in the main directory which also contains the **manage.py** and **README.rst**.

First create the log file:

```
touch appstore.log
```

Apache:

Then give your web server access to it:

```
sudo chown www-data:www-data appstore.log
```

Afterwards restart your web server:

```
sudo systemctl restart apache2
```

2.1.14 Configure Social Logins

Once the App Store is up and running social login needs to be configured. The App Store uses django-allauth for local and social login. In order to configure these logins, most providers require you to register your app beforehand.

GitHub

GitHub is currently the only supported social login. In order to register the App Store, go to your application settings page and enter the following details:

- **Application name**: Nextcloud App Store
- **Homepage URL**: https://apps.nextcloud.com
- **Authorization callback URL**: https://apps.nextcloud.com/github/login/callback/

Afterwards your **client id** and **client secret** are displayed. These need to be saved inside the database. To do that, either log into the admin interface, change your site's domain and add GitHub as a new social application or run the following command:

```
python manage.py setupsocial --github-client-id "CLIENT_ID" --github-secret "SECRET" -
↪-domain apps.nextcloud.com
```

Note: The above mentioned domains need to be changed if you want to run the App Store on a different server.

Note: For local testing use localhost:8000 as domain name. Furthermore the confirmation mail will also be printed in your shell that was used to start the development server.

2.1.15 Sync Nextcloud Releases from GitHub

The App Store needs to know about Nextcloud versions because:

- app releases are grouped by app version on the app detail page
- you can *access a REST API to get all available versions*

Before **3.2.0** releases were imported either manually or via the a shipped JSON file. This process proved to be very tedious. In **3.2.0** a command was introduced to sync releases (git tags) directly from GitHub.

The GitHub API now requires you to be authenticated so you need to obtain and configure a GitHub OAuth2 token before you run the sync command.

After obtaining the token from GitHub, add it anywhere in your settings file (**nextcloudappstore/settings/production.py**), e.g.:

```
GITHUB_API_TOKEN = '4bab6b3dfeds8857371a48855dse87d38d4b7e65'
```

You can run the command by giving it the oldest supported Nextcloud version:

```
python manage.py syncnextcloudreleases --oldest-supported="12.0.0"
```

All existing versions prior to this release will be marked as not having a release, new versions will be imported and the latest version will be marked as current version.

You can also do a test run and see what kind of versions would be imported:

```
python manage.py syncnextcloudreleases --oldest-supported="12.0.0" --print
```

To automate syncing you might want to add the command as a cronjob and schedule it every hour.

Note: Only one sync command should be run at a time, otherwise race conditions might cause unpredictable results. To ensure this use a proper cronjob daemon that supports running only one command at a time, for instance SystemD timers

Note: If run the command outside of your virtual environment you need to prefix the full path to the desired Python executable, e.g.

```
venv/bin/python manage.py syncnextcloudreleases --oldest-supported="12.0.0"
```

2.1.16 Keeping Up To Date

Updating an instance is scripted in **scripts/maintenance/update.sh**. Depending on your distribution you will have to adjust the scripts contents.

For Ubuntu you can run the provided script:

```
git pull --rebase origin master
sudo chown -R $(whoami):users /var/www
bash scripts/maintenance/update.sh apache
sudo chown -R www-data:www-data /var/www
```

Note: The above commands assume that your static content is located in **/var/www**

2.1.17 Monitoring

By default monitoring the application via New Relic is supported by simply placing a file called **newrelic.ini** into the base folder (the folder that also contains the **manage.py** file).

2.2 Store Production Install (Docker)

The App Store can be installed using Docker. This is still work in progress and therefore not recommended for production.

2.2.1 Benefits and Drawbacks

Docker is just another technology with its own upsides and downsides. Make an educated decision on whether you want to use it or go for an alternative.

Benefits:

- No need to install development libraries on your server (e.g. C compiler, Node.js)
- No need to install a specific Python version
- No need to manually run update commands. Just download a new container version and start it
- Ability to run any operating system that supports your required Docker features
- Faster deployment because Python libraries, JavaScript libraries and translations come pre-built
- Easier backups since code is completely split from production data and configuration

Drawbacks:

- Knowledge of docker-compose required to change and optimize deployment
- Docker daemon must run as root
- Python bugfixes and security updates are not available through your package manager; they require a container rebuild and deployment
- Much more complex setup due to another layer of abstraction

2.2.2 General Information

This page will detail a setup with the following configuration

- the host runs Ubuntu 16.04
- PostgreSQL and Nginx are run on the host

If you want to run a different setup you need to provide your own **docker-compose.yml** and adjust your settings accordingly.

2.2.3 Building the Image

To build the Docker image install Git, Docker and docker-compose on your development machine, e.g.:

```
sudo apt-get install docker docker-compose git
```

and start the daemon:

```
sudo systemctl enable docker
sudo systemctl restart docker
```

Then clone the repository and build the image:

```
git clone https://github.com/nextcloud/appstore
cd appstore
git checkout tags/VERSION
sudo docker-compose build production
```

Export your container to be able to upload it to your production server:

```
sudo docker save -o nextcloudappstore.tar.gz appstore_production
```

Note: Docker Hub integration would be nice

2.2.4 Deploying the Image

Upload the **nextcloudappstore.tar.gz** archive onto your production server.

2.2.5 Initial Setup

These steps are only required for your initial setup.

Install Docker and docker-compose on your production server, e.g.:

```
sudo apt-get install docker docker-compose
```

and start the daemon:

```
sudo systemctl enable docker
sudo systemctl restart docker
```

Then create and change into your target installation directory, e.g.:

```
sudo mkdir -p /srv
cd /srv
```

and create a **config/** folder which is going to hold your App Store configuration files:

```
sudo mkdir -p config/
```

Next create empty files inside the config folder:

```
sudo touch config/__init__.py
sudo touch config/production.py
sudo touch config/uwsgi.ini
sudo touch config/newrelic.ini  # only needed if you run New Relic
```

Configuring uWSGI

uWSGI is a multi language app server which will be used to run the App Store's Python code inside the container. In addition to uWSGI you will need to configure an additional web-server. A web-server is required to:

- serve static files to the client (e.g. CSS, JavaScript, images)
- encrypt the traffic with TLS

- redirect incoming requests to uWSGI

The following config file represents a minimal uwsgi config

```
[uwsgi]
chdir = /srv
wsgi-file = /srv/nextcloudappstore/wsgi.py
master = true
processes = 10
vacuum = true
uid = nextcloudappstore
gid = nextcloudappstore
socket = 0.0.0.0:8000
```

If your server does not support the uWSGI protocol natively, replace **socket** with:

```
http = 0.0.0.0:8000
```

You may also want to configure statistics and adjust threads/processes to whatever works best on your server. Consult the documentation for more information.

Configuring The App Store

The **production.py** contains all App Store specific settings that you may want to adjust:

```
from nextcloudappstore.settings.base import *

# DEBUG must be false to not leak sensitive content
DEBUG = False

# generate the SECRET_KEY by yourself for instance by using the following command:
# env LC_CTYPE=C tr -dc "a-zA-Z0-9-_\$\?" < /dev/urandom | head -c 64; echo
SECRET_KEY = 'change this!'

ALLOWED_HOSTS = ['production-domain.com']

# E-Mail settings which are used to send mails (e.g. confirm account messages)
# for more configuration options consult the Django documentation
# https://docs.djangoproject.com/en/1.11/ref/settings/#std:setting-EMAIL_HOST
DEFAULT_FROM_EMAIL = 'admin@yourdomain.com'
ADMINS = [('Your Name', 'your-mail@example.com')]
EMAIL_HOST = 'localhost'

# postgres or other db if needed if anything other than sqlite is used
# you need to create the database, user and password first
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'nextcloudappstore',
        'USER': 'nextcloudappstore',
        'PASSWORD': 'password',
        'HOST': '172.17.0.1',
        'PORT': '5432',
    }
}

# The following lines are HTTPS only!
```

(continues on next page)

(continued from previous page)

```
CSRF_COOKIE_SECURE = True
SESSION_COOKIE_SECURE = True
SECURE_HSTS_SECONDS = 31536000
SECURE_HSTS_INCLUDE_SUBDOMAINS = True
ACCOUNT_DEFAULT_HTTP_PROTOCOL = 'https'
CSP_IMG_SRC = ('https:',)

# Public and private keys for Googles recaptcha
RECAPTCHA_PUBLIC_KEY = 'YOUR PUBLIC KEY'
RECAPTCHA_PRIVATE_KEY = 'YOUR PRIVATE KEY'

LOG_FILE = join(BASE_DIR, 'logs/appstore.log')
LOG_LEVEL = 'ERROR'
LOGGING['handlers']['file']['filename'] = LOG_FILE
LOGGING['handlers']['file']['level'] = LOG_LEVEL
LOGGING['loggers']['django']['level'] = LOG_LEVEL

# Discourse user that is allowed to create categories. This will be used
# to automatically create categories when registering apps
DISCOURSE_USER = 'tom'
DISCOURSE_TOKEN = 'a token'

#########################
# Overridable Defaults: #
#########################

# Url for serving non user uploaded files like CSS, JavaScript and images
# STATIC_URL = '/static/'

# Url or domain for serving user uploaded files
# MEDIA_URL = '/media/'

# how many times a user is allowed to call the app upload route per day
# REST_FRAMEWORK['DEFAULT_THROTTLE_RATES']['app_upload'] = '100/day'
# how many times a user is allowed to call the app register route per day
# REST_FRAMEWORK['DEFAULT_THROTTLE_RATES']['app_register'] = '100/day'

# Only set this parameter if you want to use a different tmp directory for app␣
↪downloads
# RELEASE_DOWNLOAD_ROOT = '/tmp'

# minimum number of comments to calculate a rating
# RATING_THRESHOLD = 5

# number of days to include from today in the recent ratings calculation
# RATING_RECENT_DAY_RANGE = 90

# MAX_DOWNLOAD_FILE_SIZE = 1024 ** 2  # bytes
# MAX_DOWNLOAD_TIMEOUT = 60  # seconds
# MAX_DOWNLOAD_REDIRECTS = 10
# MAX_DOWNLOAD_SIZE = 20 * (1024 ** 2)  # bytes
# ARCHIVE_FOLDER_BLACKLIST = {
#     'No .git directories': r'\.git$'
# }

# DISCOURSE_URL = 'https://help.nextcloud.com'
```

(continues on next page)

(continued from previous page)

```
# If given a sub category will be created at this location
# If not given a root category will be created
# You can get the category id here at the /categories.json route, e.g.
# https://help.nextcloud.com/categories.json
# DISCOURSE_PARENT_CATEGORY_ID = 26
```

Setting Up Your Database

Install PostgreSQL on your host machine:

```
sudo apt-get install postgresql
```

To allow the container to connect to it open **/etc/postgresql/9.5/main/postgresql.conf** and modify/add the following section:

```
listen_addresses = '127.0.0.1,172.17.0.1'
```

Then whitelist your container IP in **/etc/postgresql/9.5/main/pg_hba.conf**:

```
host    nextcloudappstore nextcloudappstore 172.17.0.2/32       md5
```

Note: This expects the database user and database to be named **nextcloudappstore**, your container IP to be **172.17.0.2** and host to run on **172.17.0.1**

Then enable and start it:

```
sudo systemctl enable postgresql.service
sudo systemctl restart postgresql.service
```

and create a user and database:

```
sudo -s
su - postgres
psql
CREATE USER nextcloudappstore WITH PASSWORD 'password';
CREATE DATABASE nextcloudappstore OWNER nextcloudappstore;
\q
exit
```

Note: Use your own password instead of the password example!

Configuring Your Web-Server

First install nginx:

```
sudo apt-get install nginx
```

Then create a new configuration for it in **/etc/nginx/sites-available/nextcloudappstore**:

```
upstream nextcloudappstore {
    server 127.0.0.1:8000;
}

server {
    listen 80 default_server;
    listen [::]:80 default_server;

    # Redirect all HTTP requests to HTTPS with a 301 Moved Permanently response.
    return 301 https://$host$request_uri;
}

server {
    listen 443 ssl http2;
    listen [::]:443 ssl http2;
    server_name apps.nextcloud.com;
    charset     utf-8;

    # replace this with your ssl certificates
    ssl_certificate /etc/nginx/ssl/nextcloudappstore.crt;
    ssl_certificate_key /etc/nginx/ssl/nextcloudappstore.key;
    ssl_session_timeout 1d;
    ssl_session_cache shared:SSL:50m;
    ssl_session_tickets off;
    ssl_protocols TLSv1.2;
    ssl_ciphers 'ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-
↪ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:ECDHE-ECDSA-AES128-GCM-
↪SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-
↪SHA384:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256';
    ssl_prefer_server_ciphers on;
    ssl_prefer_server_ciphers on;
    ssl_stapling on;
    ssl_stapling_verify on;
    ssl_trusted_certificate /etc/ssl/private/ca-certs.pem;

    add_header Strict-Transport-Security max-age=15768000;
    add_header X-Content-Type-Options nosniff;
    add_header X-XSS-Protection "1; mode=block";

    client_max_body_size 75M;
    location /media  {
        alias /srv/media;
    }

    location /static {
        alias /srv/static;
    }

    location / {
        uwsgi_pass nextcloudappstore;
        include uwsgi_params;
    }
}
```

Finally replace your default configuration:

```
sudo rm /etc/nginx/sites-enabled/default
```

(continues on next page)

(continued from previous page)

```
sudo ln -s /etc/nginx/sites-available/nextcloudappstore /etc/nginx/sites-enabled/
↪default
sudo systemctl enable nginx
sudo systemctl restart nginx
```

Configuring New Relic (Optional)

TBD

Creating Docker-Compose Configuration

Either create your own configuration or grab a copy of our docker-compose.yml and modify it if necessary. Place the file in your designated directory:

```
cd /srv
sudo wget https://raw.githubusercontent.com/nextcloud/appstore/master/docker-compose.
↪yml
```

Starting the Image

First load the latest uploaded image:

```
sudo docker load -i /path/to/nextcloudappstore.tar.gz
```

Then change into your server directory and start the container:

```
cd /srv
sudo docker-compose up production
```

The following directories will be created initially:

- **static**: holds read only files which need to be served by your web-server
- **media**: holds user uploaded files
- **logs**: contains your log file

The **static** directory will be populated with static files when a container is started and all database migrations and fixtures will be imported.

Creating an Admin User

To create the initial admin user and verify his email, run the following command:

```
sudo docker-compose exec production python manage.py createsuperuser --username admin␣
↪--email admin@admin.com
sudo docker-compose exec production python manage.py verifyemail --username admin --
↪email admin@admin.com
```

The first command will ask for the password.

Configure Social Logins

Once the App Store is up and running social login needs to be configured. The App Store uses django-allauth for local and social login. In order to configure these logins, most providers require you to register your app beforehand.

GitHub

GitHub is currently the only supported social login. In order to register the App Store, go to your application settings page and enter the following details:

- **Application name**: Nextcloud App Store
- **Homepage URL**: https://apps.nextcloud.com
- **Authorization callback URL**: https://apps.nextcloud.com/github/login/callback/

Afterwards your **client id** and **client secret** are displayed. These need to be saved inside the database. To do that, either log into the admin interface, change your site's domain and add GitHub as a new social application or run the following command:

```
sudo docker-compose exec python manage.py setupsocial --github-client-id "CLIENT_ID" -
↪-github-secret "SECRET" --domain apps.nextcloud.com
```

Note: The above mentioned domains need to be changed if you want to run the App Store on a different server.

Note: For local testing use localhost:8000 as domain name. Furthermore the confirmation mail will also be printed in your shell that was used to start the development server.

Sync Nextcloud Releases from GitHub

The App Store needs to know about Nextcloud versions because:

- app releases are grouped by app version on the app detail page
- you can *access a REST API to get all available versions*

Before **3.2.0** releases were imported either manually or via the a shipped JSON file. This process proved to be very tedious. In **3.2.0** a command was introduced to sync releases (git tags) directly from GitHub.

You can run the command by giving it the oldest supported Nextcloud version:

```
sudo docker-compose exec python manage.py syncnextcloudreleases --oldest-supported=
↪"12.0.0"
```

All existing versions prior to this release will be marked as not having a release, new versions will be imported and the latest version will be marked as current version.

You can also do a test run and see what kind of versions would be imported:

```
sudo docker-compose exec python manage.py syncnextcloudreleases --oldest-supported=
↪"12.0.0" --print
```

The GitHub API is rate limited to 60 requests per second. Depending on how far back your **oldest-supported** version goes a single command might fetch multiple pages of releases. If you want to run the command more than 10 times per hour it is recommended to obtain and configure a GitHub OAuth2 token.

After obtaining the token from GitHub, add it anywhere in your settings file (**production.py**), e.g.:

```
GITHUB_API_TOKEN = '4bab6b3dfeds8857371a48855d3e87d38d4b7e65'
```

To automate syncing you might want to add the command as a cronjob and schedule it every hour.

Note: Only one sync command should be run at a time, otherwise race conditions might cause unpredictable results. To ensure this use a proper cronjob daemon that supports running only one command at a time, for instance SystemD timers

2.3 Store Upgrade Notices

2.3.1 3.2.0

Configuring Nextcloud Release Sync

3.2.0 changed the way Nextcloud releases are managed in the App Store. Instead of manually adjusting releases in the admin interface or importing updated fixtures via JSON files releases are now synced directly from GitHub.

Consult *Sync Nextcloud Releases from GitHub* or if you are using Docker *Sync Nextcloud Releases from GitHub* in order to run or configure the release sync command.

CHAPTER 3

App Store Developer Documentation

Look here if you want to work on the app store's source code

3.1 Store Development Installation

This setup details a local development installation in order to work and test App Store changes. The App Store is build using Django. The frontend is written in TypeScript and does not yet use a JavaScript framework; this decision might change however depending on how JavaScript intensive things might become.

Note: Only use this guide for your local development installation which is **not connected to the Internet** since your installation will be be initialized with insecure defaults!

3.1.1 Installing Packages

First you want to switch your machine to an up to date Node.js version and install Yarn:

```
curl -sS https://deb.nodesource.com/gpgkey/nodesource.gpg.key | sudo apt-key add -
echo "deb https://deb.nodesource.com/node_8.x xenial main" | sudo tee /etc/apt/
↪sources.list.d/nodesource.list
echo "deb-src https://deb.nodesource.com/node_8.x xenial main" | sudo tee -a /etc/apt/
↪sources.list.d/nodesource.list

curl -sS https://dl.yarnpkg.com/debian/pubkey.gpg | sudo apt-key add -
echo "deb https://dl.yarnpkg.com/debian/ stable main" | sudo tee /etc/apt/sources.
↪list.d/yarn.list
```

Then install the following libraries:

```
sudo apt-get update
sudo apt-get install python3-venv python3-wheel libxslt-dev libxml2-dev libz-dev␣
↪libpq-dev build-essential python3-dev python3-setuptools git gettext libssl-dev␣
↪libffi-dev nodejs yarn
```

(continues on next page)

(continued from previous page)

```
```

Finally download the latest geckodriver and install it (adjust URLs/package names if needed):

```
wget https://github.com/mozilla/geckodriver/releases/download/v0.21.0/geckodriver-v0.
↪21.0-linux64.tar.gz
sudo sh -c 'tar -x geckodriver -zf geckodriver-v0.21.0-linux64.tar.gz -O > /usr/bin/
↪geckodriver'
sudo chmod +x /usr/bin/geckodriver
rm geckodriver-v0.21.0-linux64.tar.gz
```

3.1.2 Download the Source

Clone the repository using git and change into it:

```
git clone https://github.com/nextcloud/appstore.git
cd appstore
```

3.1.3 App Store Setup

The project root contains a **Makefile** which allows you to quickly set everything up by running:

```
make dev-setup
```

This will automatically set up the web app using **venv**, **SQLite** as database and create a default **development** settings file in **nextcloudappstore/settings/development.py**. You need to review the development settings and change them according to your setup. An admin user with name **admin** and password **admin** will also be created.

3.1.4 Launching the Development Server

The server can be started after activating the virtual environment first:

```
source venv/bin/activate
export DJANGO_SETTINGS_MODULE=nextcloudappstore.settings.development
python manage.py runserver
```

The website is available at http://127.0.0.1:8000. Code changes will auto reload the server so happy developing! For more documentation on development, check out *Store Development*

Every time you start a new terminal session you will need to reactive the virtual environment and set the development settings:

```
source venv/bin/activate
export DJANGO_SETTINGS_MODULE=nextcloudappstore.settings.development
```

We therefore recommend creating a small bash alias in your **~/.bashrc**:

```
alias cda='cd path/to/appstore && source venv/bin/activate && export DJANGO_SETTINGS_
↪MODULE=nextcloudappstore.settings.development'
```

3.1.5 Keeping Up to Date

Note: Before updating it is recommended to stop the development server.

To check out the latest changes simply pull:

```
git pull --rebase origin master
```

then make sure that the virtual environment is enabled and install new libraries:

```
make update-dev-deps
```

apply new database migrations:

```
python manage.py migrate
```

and build the latest frontend:

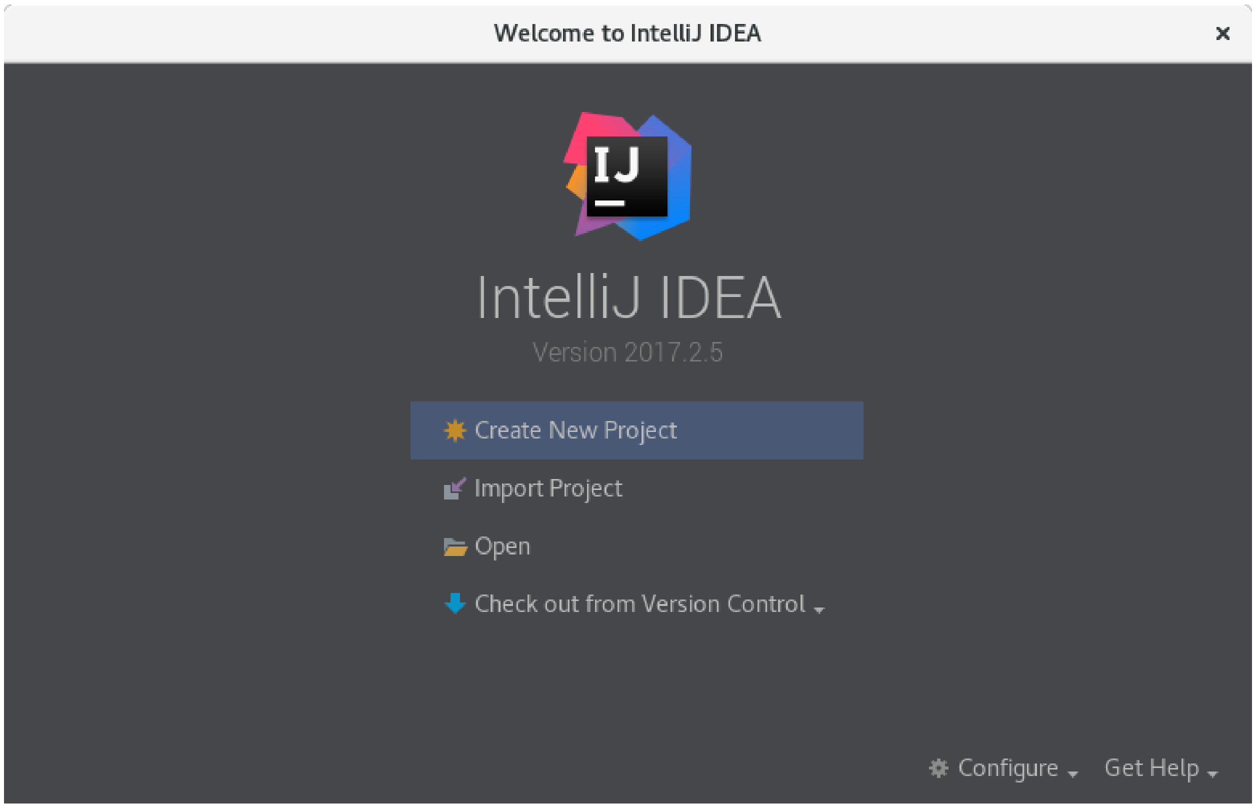
```
yarn run build
```

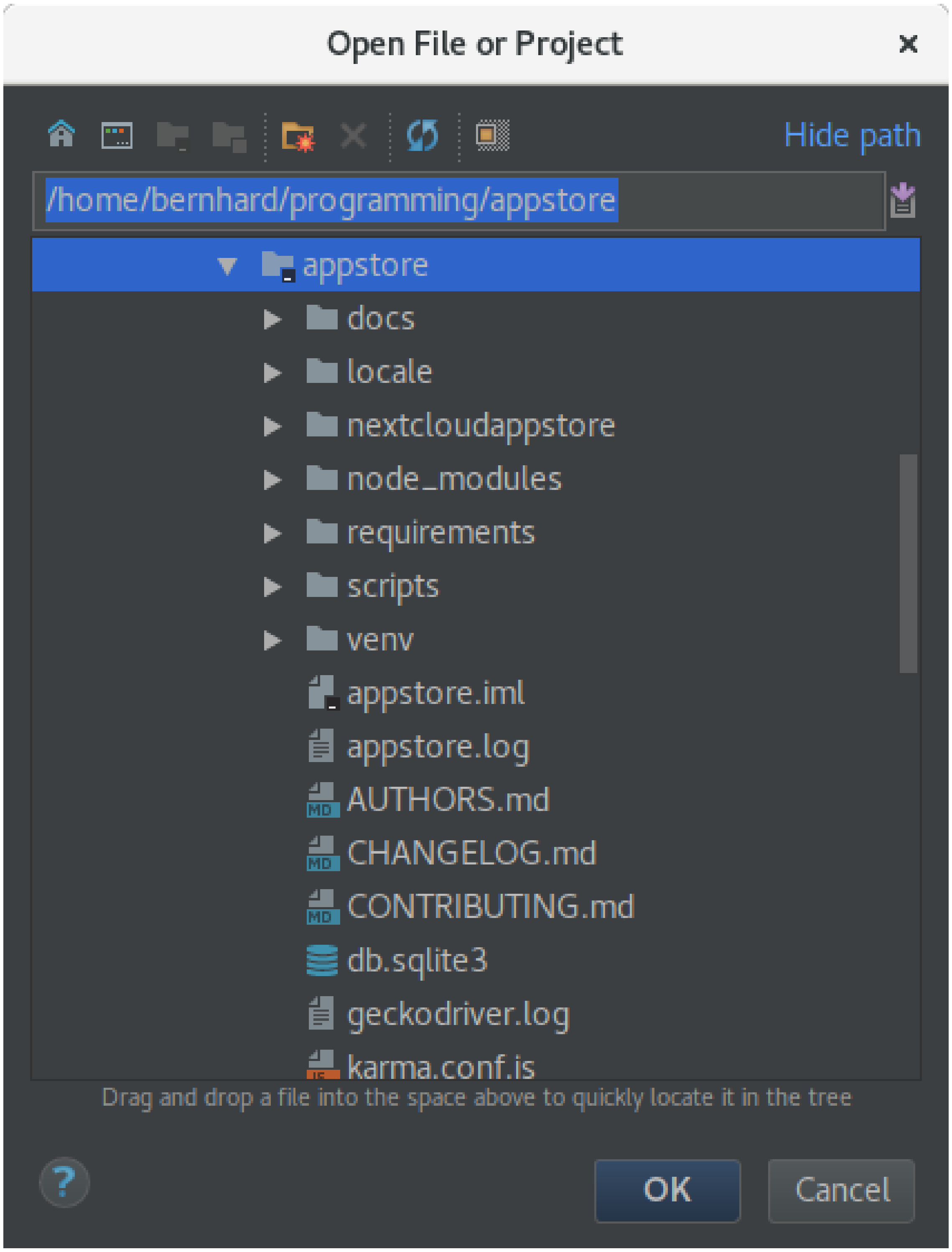
3.1.6 IntelliJ IDEA Ultimate/PyCharm Setup

You can use any editor/IDE that you want to. However if you are already using IntelliJ IDEA Ultimate with the Python plugin or PyCharm you can import a fully installed project with the following steps:

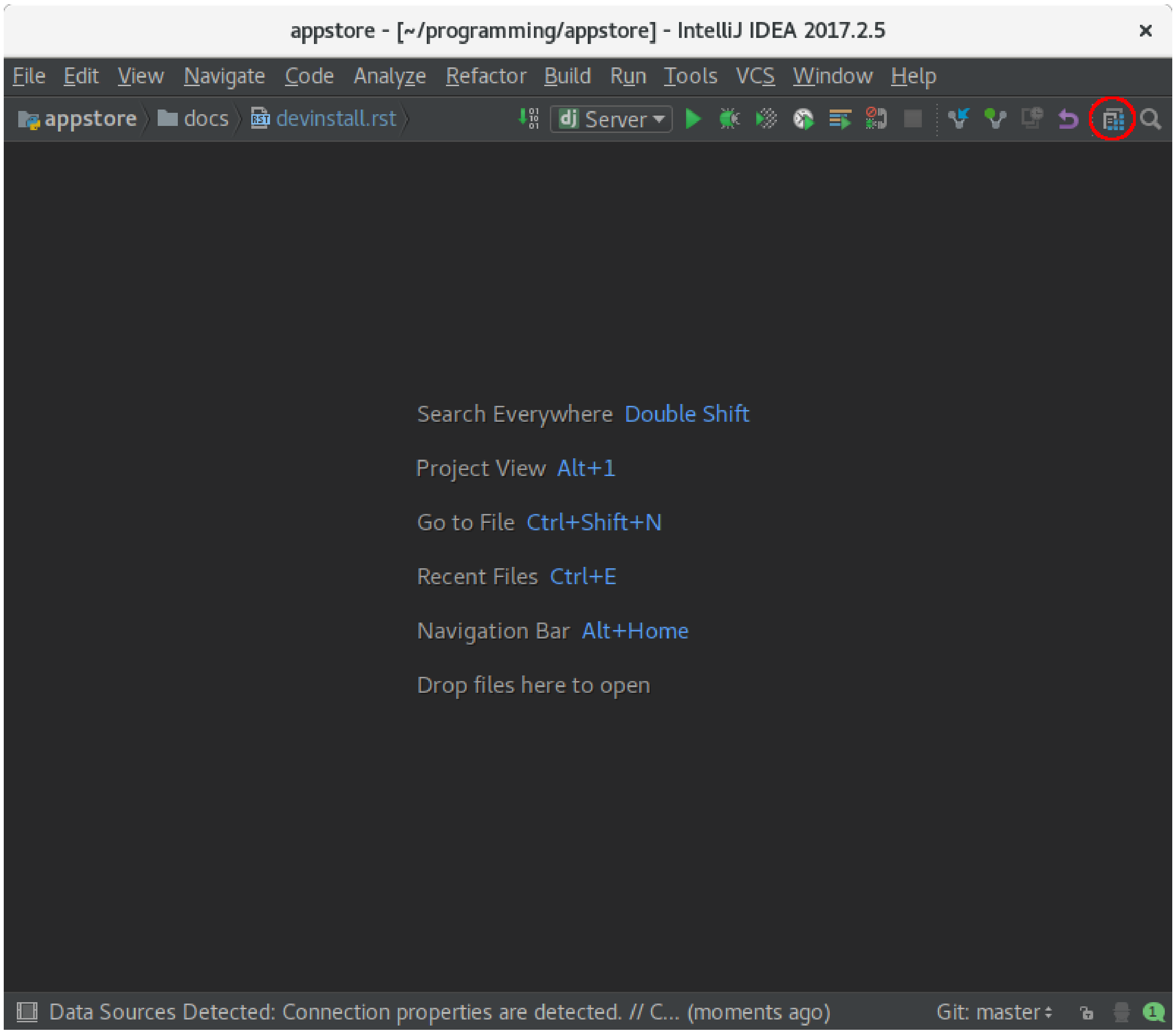
Note: The configuration uses the **Django** Facet which is only included in the commercial IDE releases.

In the project window click **Open** and select the cloned **appstore** folder:

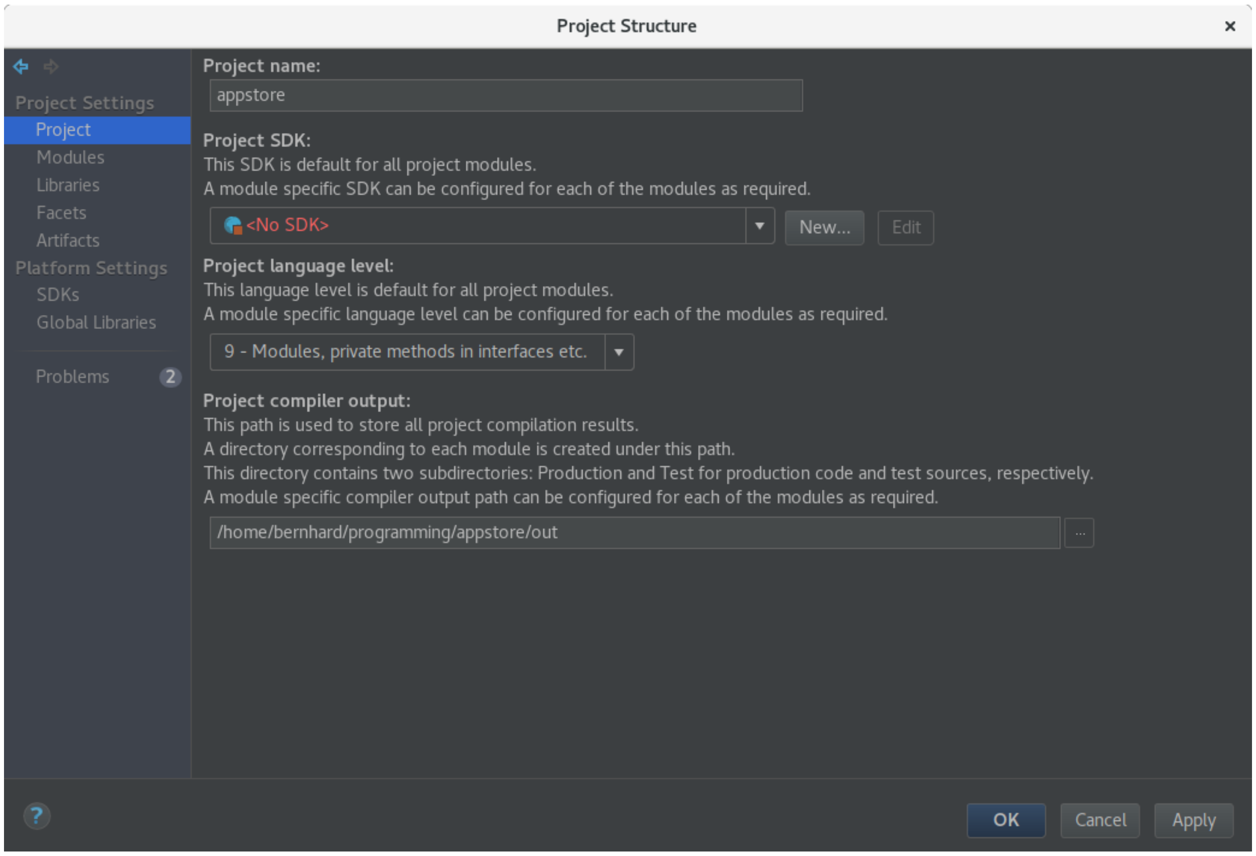

Welcome to IntelliJ IDEA
IntelliJ IDEA
Version 2017.2.5
Create New Project
Import Project
Open
Check out from Version Control
Configure
Get Help



Afterwards you are presented with the main window. From here, click on the project settings icon:


appstore - [~/programming/appstore] - IntelliJ IDEA 2017.2.5
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
appstore docs devinstall.rst
Server
Search Everywhere Double Shift
Project View Alt+1
Go to File Ctrl+Shift+N
Recent Files Ctrl+E
Navigation Bar Alt+Home
Drop files here to open
Data Sources Detected: Connection properties are detected. // C... (moments ago)
Git: master

Then you need to add a new SDK. To do that click on **New…** -> **Python SDK** -> **Add Local** and select the Python executable **venv/bin/python** in your local **venv** folder:

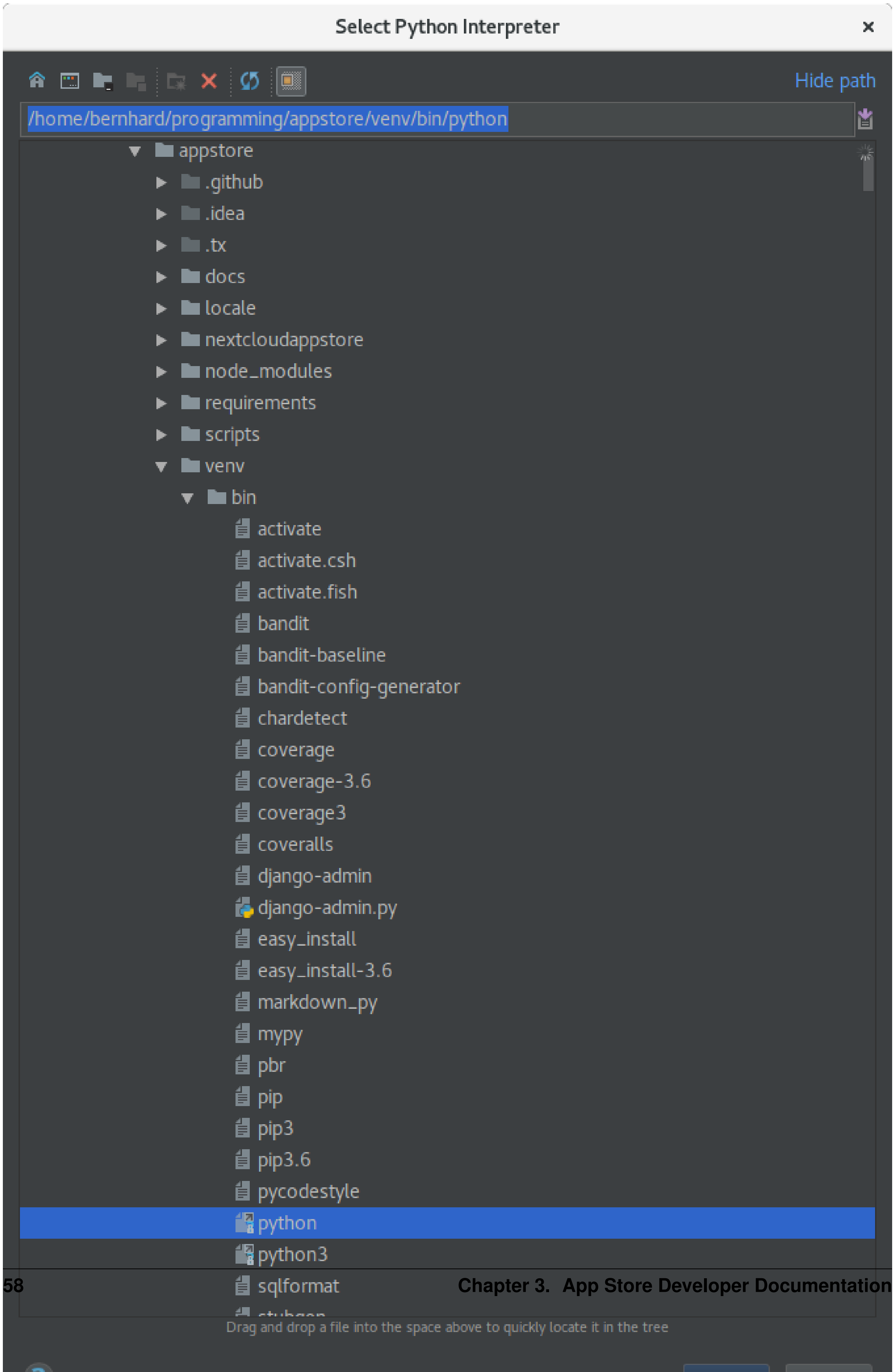


Project name:
appstore

Project SDK:
This SDK is default for all project modules.
A module specific SDK can be configured for each of the modules as required.
<No SDK>

Project language level:
This language level is default for all project modules.
A module specific language level can be configured for each of the modules as required.
9 - Modules, private methods in interfaces etc.

Project compiler output:
This path is used to store all project compilation results.
A directory corresponding to each module is created under this path.
This directory contains two subdirectories: Production and Test for production code and test sources, respectively.
A module specific compiler output path can be configured for each of the modules as required.
/home/bernhard/programming/appstore/out

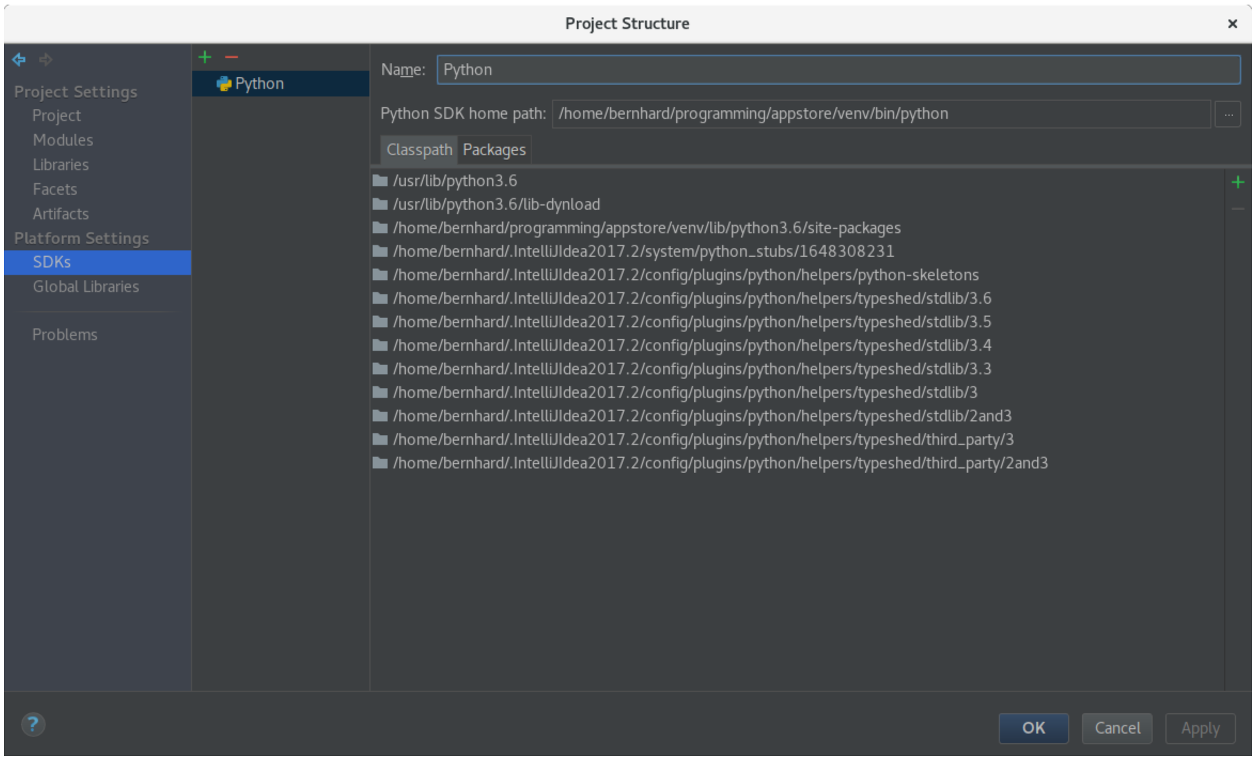


```
/home/bernhard/programming/appstore/venv/bin/python
```

```
appstore
  .github
  .idea
  .tx
  docs
  locale
  nextcloudappstore
  node_modules
  requirements
  scripts
  venv
    bin
      activate
      activate.csh
      activate.fish
      bandit
      bandit-baseline
      bandit-config-generator
      chardetect
      coverage
      coverage-3.6
      coverage3
      coveralls
      django-admin
      django-admin.py
      easy_install
      easy_install-3.6
      markdown_py
      mypy
      pbr
      pip
      pip3
      pip3.6
      pycodestyle
      python
      python3
      sqlformat
```

Drag and drop a file into the space above to quickly locate it in the tree

In your project settings go to **SDKs** and select your recently added Python SDK. Rename it to **Python** and close the project window by clicking **OK**.



Name: Python

Python SDK home path: /home/bernhard/programming/appstore/venv/bin/python

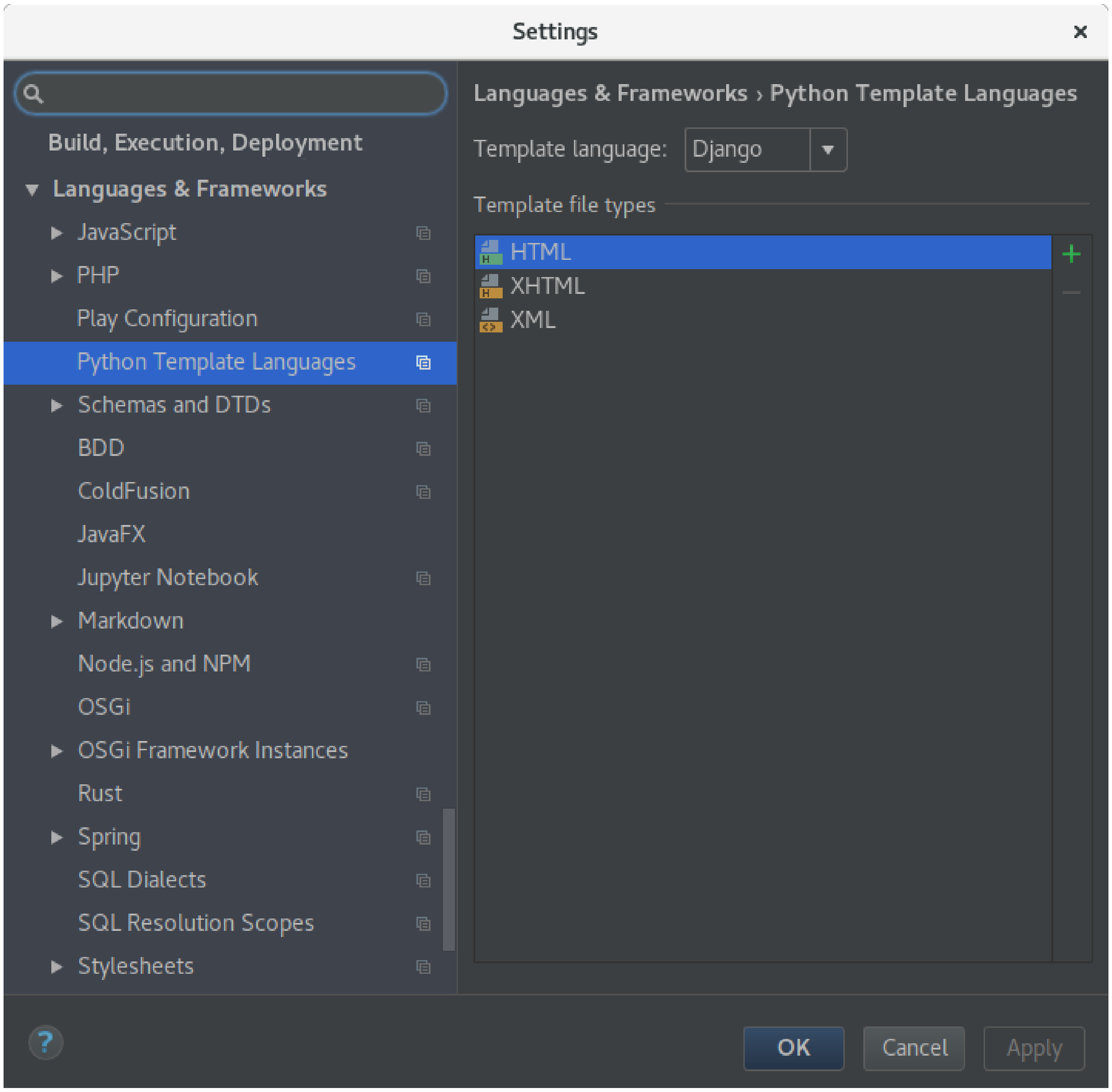
```
/usr/lib/python3.6
/usr/lib/python3.6/lib-dynload
/home/bernhard/programming/appstore/venv/lib/python3.6/site-packages
/home/bernhard/.IntelliJIdea2017.2/system/python_stubs/1648308231
/home/bernhard/.IntelliJIdea2017.2/config/plugins/python/helpers/python-skeletons
/home/bernhard/.IntelliJIdea2017.2/config/plugins/python/helpers/typeshed/stdlib/3.6
/home/bernhard/.IntelliJIdea2017.2/config/plugins/python/helpers/typeshed/stdlib/3.5
/home/bernhard/.IntelliJIdea2017.2/config/plugins/python/helpers/typeshed/stdlib/3.4
/home/bernhard/.IntelliJIdea2017.2/config/plugins/python/helpers/typeshed/stdlib/3.3
/home/bernhard/.IntelliJIdea2017.2/config/plugins/python/helpers/typeshed/stdlib/3
/home/bernhard/.IntelliJIdea2017.2/config/plugins/python/helpers/typeshed/stdlib/2and3
/home/bernhard/.IntelliJIdea2017.2/config/plugins/python/helpers/typeshed/third_party/3
/home/bernhard/.IntelliJIdea2017.2/config/plugins/python/helpers/typeshed/third_party/2and3
```

You are now ready to start developing. You can start the server by choosing the run configuration **Server** and run all tests by selecting the run configuration **Tests**.

Note: Should you have issues with unrecognized imports just invalidate your caches by going to your global menu and clicking **File** -> **Invalidate Caches / Restart** -> **Invalidate and Restart**

IntelliJ IDEA Ultimate Specific Setup

To enable support for Django templates, open your IDE settings by clicking **File** -> **Settings**. Then go to **Languages and Frameworks** -> **Python Template Languages** and select **Django** in the drop down menu on the right. Confirm your changes by clicking **OK**



Languages & Frameworks › Python Template Languages

Template language: Django

Template file types

HTML
XHTML
XML

3.2 Store Development

The app store uses a Makefile for common tasks. The following useful Make commands are available:

make dev-setup installs a local development setup (requires previous setup, see *Store Development Installation*)

make test runs the frontend and backend test suite

make lint runs the code-style checker

make authors updates the AUTHORS.rst file based on the git database

make docs regenerates up to html date docs in docs/_build/html

make update-dev-deps updates your python, bower and yarn dependencies

make resetup kills the current sqlite database and creates a new one

make test-data downloads and sets up test apps, needs certificate validation to be disabled and a running server at http://127.0.0.1:8000 . Keep in mind that in order to display any app releases on the page you need to first sync *the available nextcloud releases* with the oldest version being **11.0.0**

make prod-data prod_version=12.0.0 similar to **make test-data** but installs all apps from production for a nextcloud version locally

make l10n compiles and installs translations

3.2.1 Frontend

The frontend is written in TypeScript and compiles to ES6 using Webpack.

To run the frontend build make sure that all your deps are up to date:

```
yarn install
```

and then run:

```
yarn run build
```

If you are developing and wish to automatically compile on filechanges run:

```
yarn run watch
```

Users

By default the following users will be ready to use:

- admin

Running **make test-data** will additionally create the following users:

- user1
- user2
- user3

All users have the same password as their username

Testing

The unit and integration test suite is run by executing the following command:

```
yarn test
```

If you are developing and wish to automatically re-run your test suite on filechanges run:

```
yarn run watch-test
```

3.3 Store Documentation

The documentation is built using Sphinx and is located inside the **docs** folder.

To change the documentation it is recommended to first install all required libraries in a virtual environment. To do that follow the guide for *Store Development Installation*.

To build the documentation change into the **docs** folder and run:

```
make html
```

Then open the file in **_build/html/index.html** in your favorite browser.

3.4 Store Translation

The App Store uses Django's translation system.

3.4.1 Generating Translations

To initially add a language for translation run:

```
python manage.py makemessages -l $CODE -i venv
```

where **$CODE** is **de** for instance. This will create the required directories and files in the **locales/** folder.

The above command only needs to be run if you want to add a new language. To update existing translation files run:

```
python manage.py makemessages -a -i venv
```

Note: The above requirements require exported environment variables and installed libraries. To find out how to do that see *Store Development Installation*.

3.4.2 Generating Database Translations

Certain translated strings like categories are stored in the database. If you change them in the database, you need to extract them into **.po** files. To do that run:

```
python manage.py create createdbtranslations
python manage.py makemessages -a -i venv
```

To import the translated messages back into the database run:

```
python manage.py compilemessages
python manage.py importdbtranslations
```

3.4.3 Deploying Translations

Each time the **.po** files are changed, they need to be compiled into a specific binary format for **gettext** using:

```
python manage.py compilemessages
```

Further details can be looked up in Django's documentation online

Afterwards you want to import all the translated content in the database:

> python manage.py importdbtranslations

3.4.4 Managing Translations

Translations are managed on Transifex. If you want to help translating the App Store, click the **Join team** button and create an issue on our issue tracker if your request takes longer than 2 days to process.

All translatable languages for the App Store can be found on its resources page. Select the language you want to translate and hit the **Translate** or **view strings online** button.

Translated content will be pulled daily from Transifex.

CHAPTER 4

Indices and tables

- genindex
- modindex
- search