

Unit 2

Architecture

Configure a Forest and Database

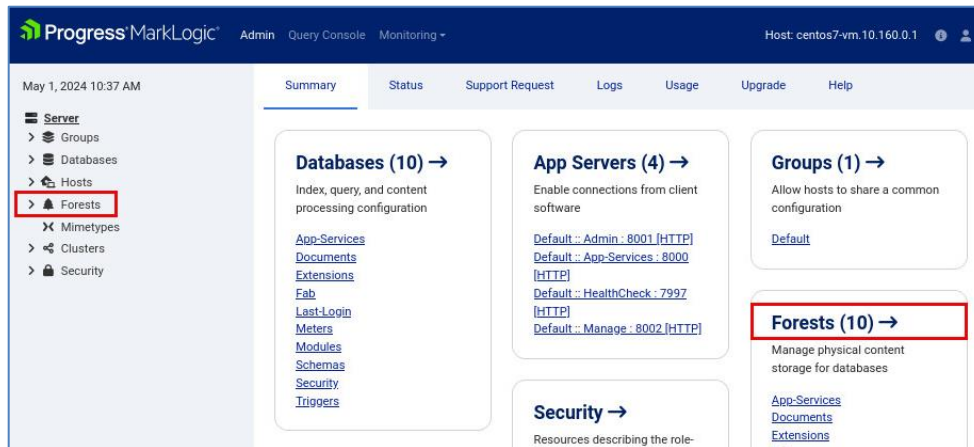
Configure an App Server

Use ml-gradle

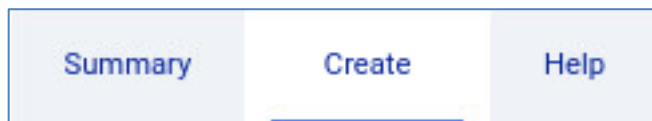
Exercise 1: Configure a Forest and Database

Note that a database is just a set of configurations. An operational database **MUST** have at least one Forest attached. The Forest is the physical storage of documents.

1. Go the Admin UI (<http://localhost:8001>)
2. Click on “Forests” using one of the two links shown below:



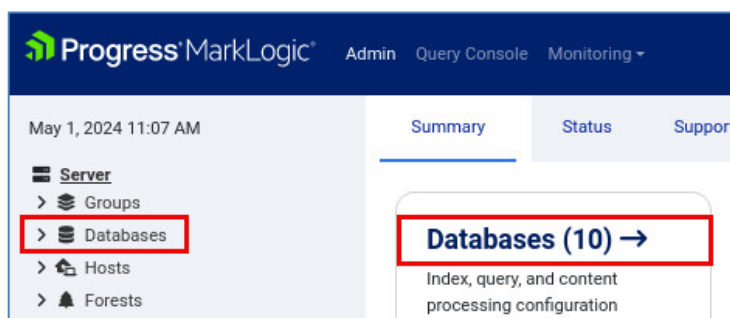
3. Click on the “Create” tab.



4. Enter `playground-1` as Forest Name and click “OK”.
 - Using numbers for suffix is a convention to indicate the number of forests attached to a database.
5. Open a Terminal window (Applications→System Tools→Mate Terminal).
6. Check the forest that got created:

```
$ ls /var/opt/MarkLogic/Forests/playground-1
```

- `/var/opt/MarkLogic/` is the default location for MarkLogic Server data in a Linux environment. This location can be overridden using the `/etc/marklogic.conf` file from the previous exercise.
7. Back in the Admin UI, click on “Databases” using one of the two links shown below:



8. Click on the “Create” tab.

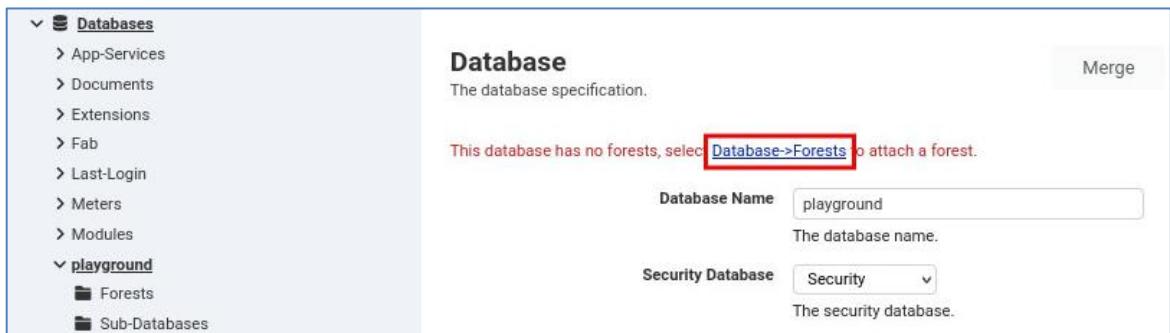


9. Enter `playground` as Database Name and click “OK”.

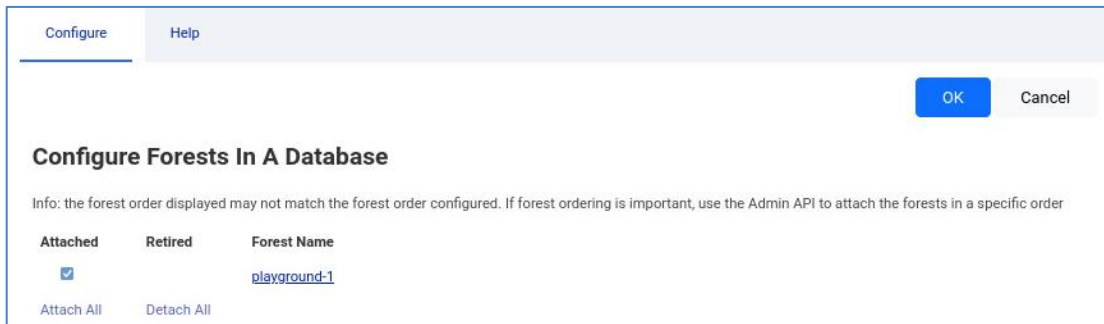
- Notice how the schema and triggers database are set to “none”. When needed, it is recommended that you create a corresponding content-specific schema and triggers database to avoid confusion between content databases.

Note: Documents are stored in a forest, while a database is just a set of configurations. To be able to use a database, you need to attach at least one forest to a database

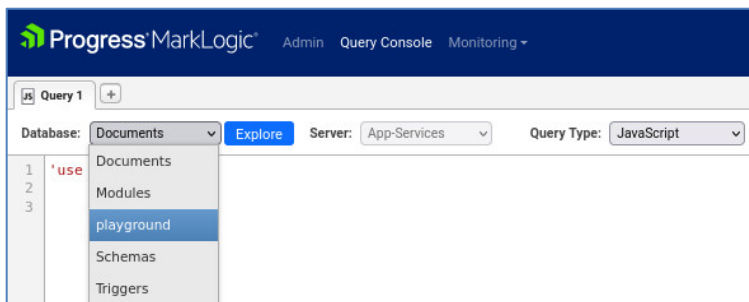
10. Click on “Database → Forests” (any as shown below):



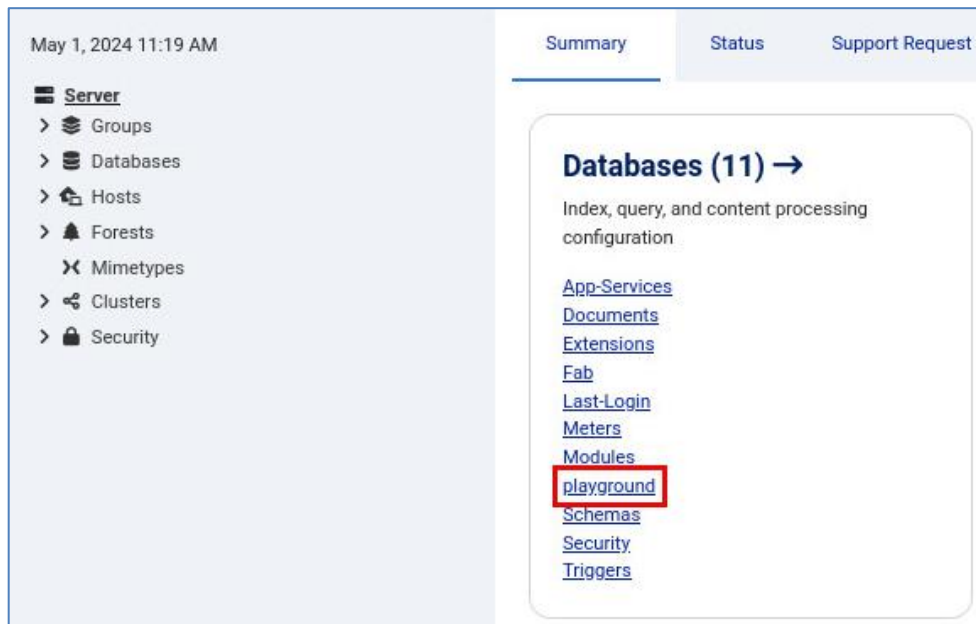
11. “check” the Attached option as shown below and click “OK”, you now have a functional database.



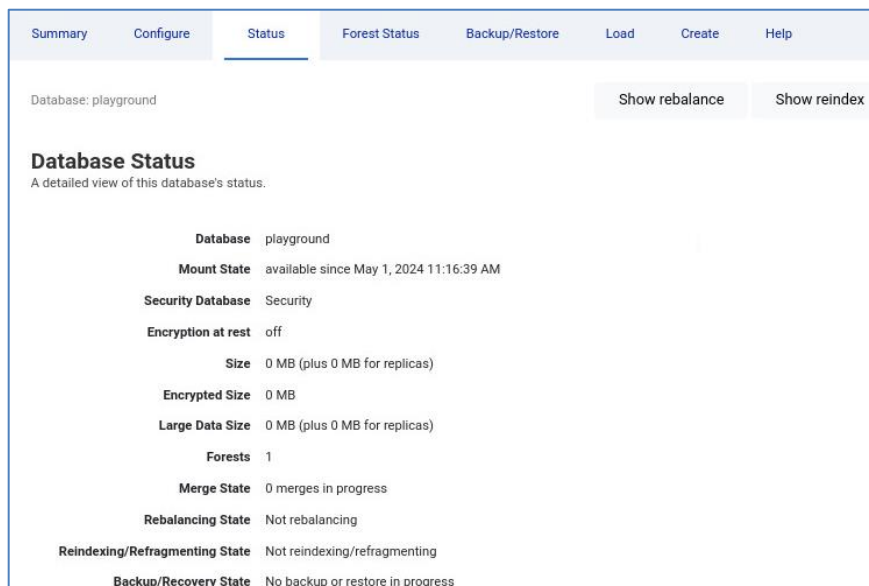
12. Open Query Console (<http://localhost:8000/qconsole>) and login as admin/admin to view this new database.



13. Back in the Admin UI, select “Server” and under Databases, click on “playground” (as shown below):



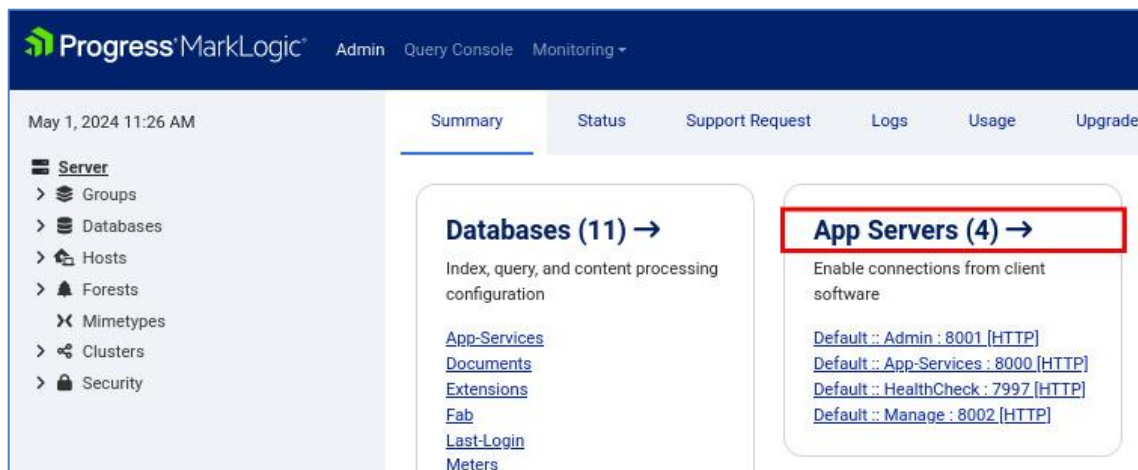
14. Click on the “Status” tab and scroll down to see the various database details.



Exercise 2: Configure an App Server

The only way to communicate with a MarkLogic database is via an App Server. There are different types of App Servers for different purposes. We will be configuring a REST API App Server, which is a specialized form of an HTTP App Server.

1. In the Admin UI (<http://localhost:8001>) click “App Servers” (as shown below):



2. Click the “Create HTTP” tab:

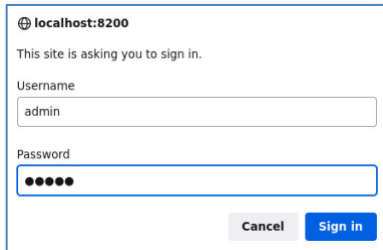


3. Use the following information and click “OK”:

Field	Value
Server Name	playground
Root	/
Port	8200
Database	playground
Error Handler (scroll to bottom of the page)	/MarkLogic/rest-api/error-handler.xqy
URL rewriter	/MarkLogic/rest-api/rewriter.xml
rewrite resolves globally	true

- Notice how “authentication” is set to “digest” by default.
- Notice how “default user” is set to “nobody” by default.

4. Open a browser tab and enter “<http://localhost:8200>” and enter admin/admin as credentials.:



localhost:8200

This site is asking you to sign in.

Username

admin

Password

•••••

Cancel Sign in

5. You should see something like the following:



MarkLogic REST Server

- Search and retrieve XML results - </v1/search?format=xml>
- Search and retrieve JSON results - </v1/search?format=json>
- Search example - </v1/search?q=&start=10&pageLength=5>
- Query Configuration - </v1/config/query>
- Transform Configuration - </v1/config/transforms>

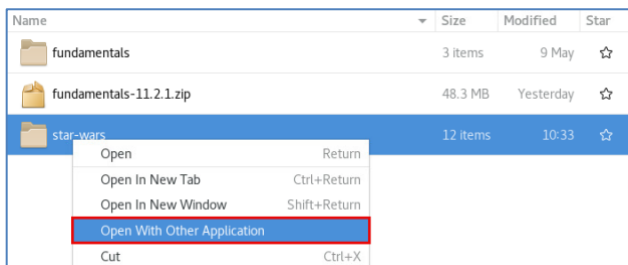
At this point, you now have an App Server that supports the use of the [MarkLogic Client API](#). See the docs for more information about these endpoints.

Exercise 3: Use ml-gradle

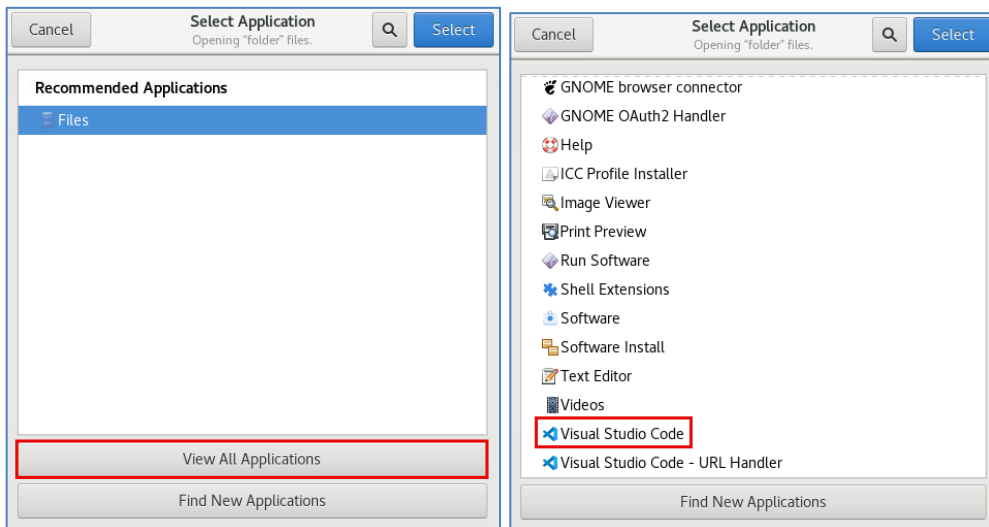
The above steps are meant to familiarize yourselves with the Admin UI. In practice, however, it is not recommended to configure your databases and app servers manually as this is prone to human error.

This exercise focuses on MarkLogic **ml-gradle**, the recommended deployment tool. This is a MarkLogic plugin to gradle that allows your configurations to be encoded, versioned and deployed across multiple MarkLogic instances/environments consistently.

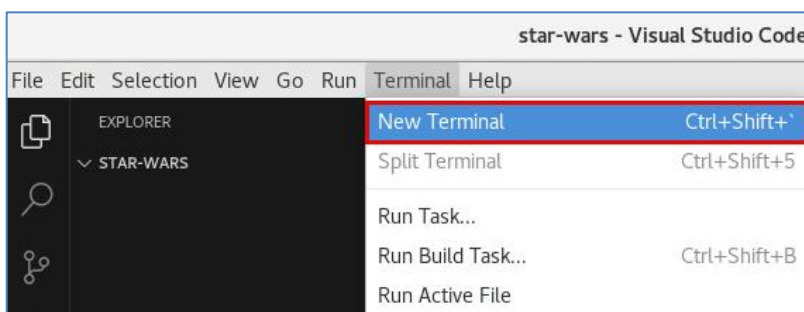
1. Open the File Browser and navigate to the 'Desktop' folder. Right click on the 'star-wars' folder and select 'Open With Other Application'.



Select 'View All Applications' button, scroll down to choose 'Visual Studio Code' and click the 'Select' button.



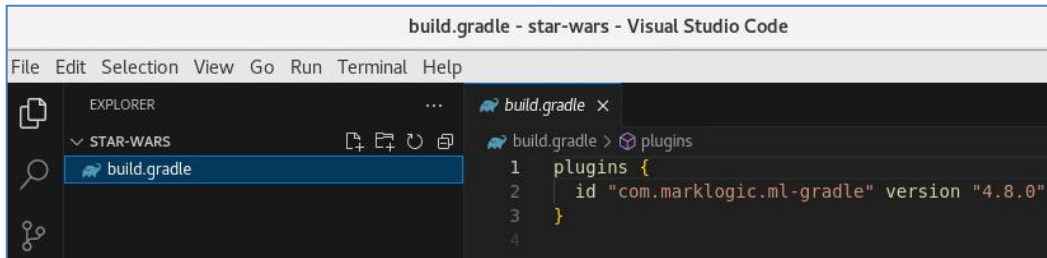
2. In the Visual Studio Code (VSC) application, from the menu select 'Terminal' and then 'New Terminal'.



3. In the Terminal window, execute the Copy statement below of the prepared 'build.gradle' file.

```
$ cp /home/cent/Desktop/fundamentals/solutions/unit_02/exercise_03/build.gradle .
```

In VSC select the build.gradle file to view the contents.



4. Run the following Gradle Task to generate boilerplate files.

```
$ gradle mlNewProject
```

You will be prompted for the following:

```
Welcome to the new project wizard. Please respond to each of the following prompts to start a
new project.

Each prompt below begins with '[ant:input]'; type your response for each prompt on the blank
line, or press 'Enter' to accept the default value or values in the brackets.

Note that this will overwrite your current build.gradle and gradle.properties files, and back
up copies of each will be made.

[ant:input] Application name: [myApp]
<-----> 0% EXECUTING [2m 15s]
> :mlNewProject
█
```

5. Enter the following values when prompted:

Question	Value
Application Name	star-wars
Host to deploy	localhost
MarkLogic admin username	admin
MarkLogic admin password	admin
REST API port	8100
Test REST API port	Leave blank for no server
Do you want support for multiple environments?	y
Do you want resource files for a content database and a set of users/roles created?	y

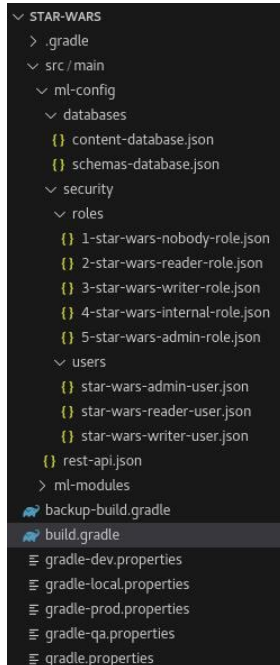
- For the empty fields above, just press the enter key to accept the default values.

View the summary of the generated documents:

```
Updating build.gradle so that the Gradle properties plugin can be applied
Writing: /home/cent/Desktop/star-wars/build.gradle
Writing: /home/cent/Desktop/star-wars/gradle.properties
Writing: /home/cent/Desktop/star-wars/gradle-dev.properties
Writing: /home/cent/Desktop/star-wars/gradle-local.properties
Writing: /home/cent/Desktop/star-wars/gradle-qa.properties
Writing: /home/cent/Desktop/star-wars/gradle-prod.properties
Making directory: /home/cent/Desktop/star-wars/src/main/ml-config
Making directory: /home/cent/Desktop/star-wars/src/main/ml-modules
```

Note: In case you missed supplying the application name and/or the port, just modify the resulting gradle.properties file accordingly.

6. The result is the content created in the `/home/cent/Desktop/star-wars` Gradle Project folder:

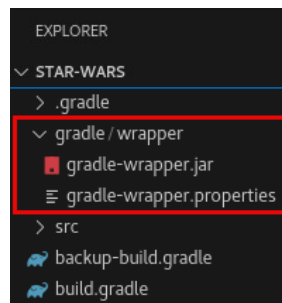


- Note: These files are common to many MarkLogic Server projects.
- Note: Gradle uses placeholders, signified by `%% var_name %%`, in these files. These allow for easy configuration using the various `gradle-<env>.properties` file. More information about these place holders are discussed in the [github wiki](#).

7. Create the gradle wrapper.

```
$ gradle wrapper
```

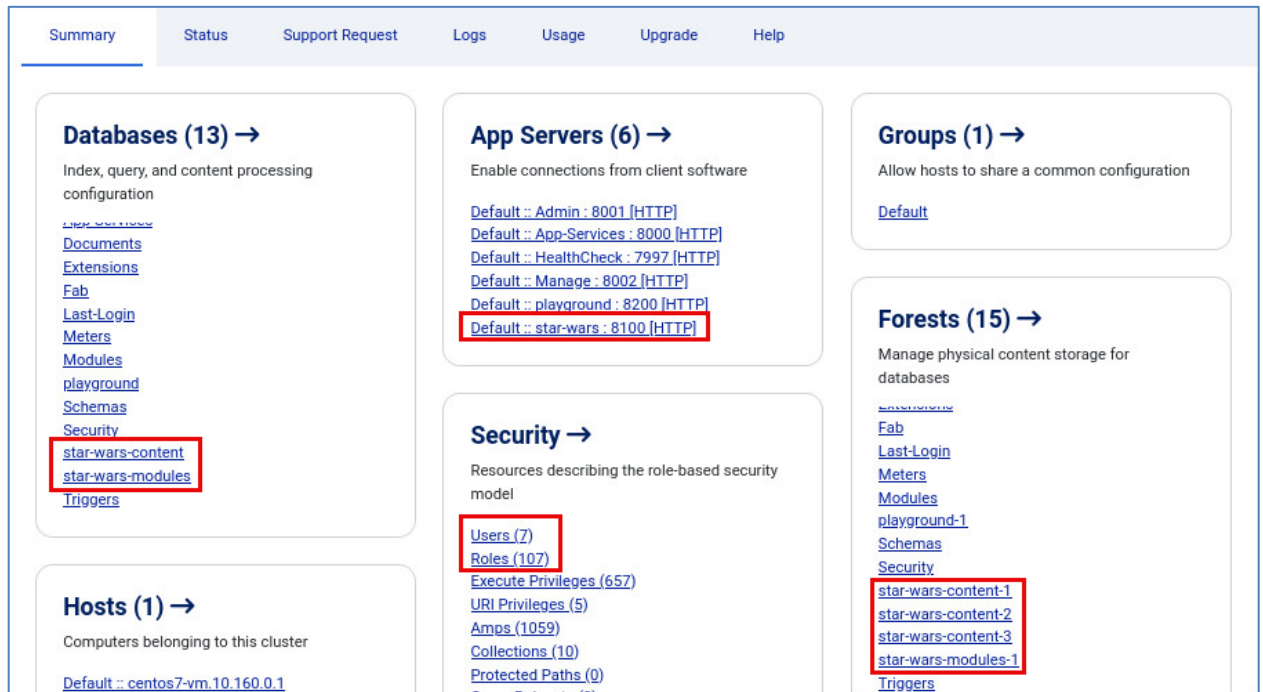
- Note the addition of the `/gradle/wrapper/` folder and files have been added to the Gradle Project folder.



- The gradle wrapper (gradlew.bat/gradlew.sh) allows the user to run gradle tasks without installing gradle system-wide.
 - The gradle wrapper also acts as a guarantee that the project-specific tasks will complete on that version of gradle.
8. Run the task to deploy our generated project.

```
$ ./gradlew mlDeploy
```

9. Open the Admin UI (<http://localhost:8001>) to see the changes take effect.
- Notice how your databases now includes a star-wars-modules database. This is considered best practice for any MarkLogic Server application.



Challenge:

To test your familiarity with ml-gradle, create a new project named `top-songs` that uses port 8300.