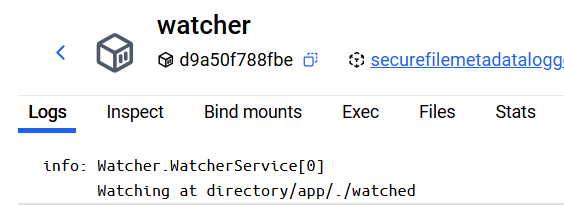
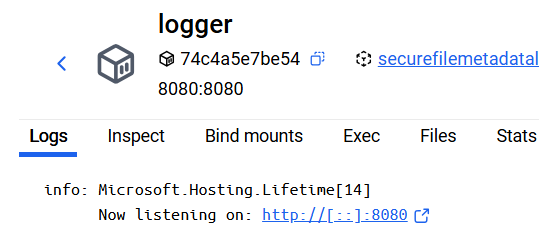
To run in Docker:  
Decompress SecureFileMetadataLogger.zip file and run docker compose from solution directory " C:\...\SecureFileMetadataLogger.  
"docker-compose up –build". It will build and run docker containers  
  
   
  
in watcher service, go to app/watched directory, enter command "echo "Test file" > test.txt"   
to create file test.txt. Look in the logs of services.  
  
  
To run on localhost:  
1.From logger directory, run command   
 " dotnet Logger.dll --urls "http://localhost:8080"  
3. From watcher directory, run command   
 dotnet Watcher.dll  
Save file in watcher/watched directory. Look in the consoles of services to see the changes.  
  
-------------------------------------------------------------------------------------------------------------------------

Design:  
Three projects. Watcher project, Logger project and Common project for DTO(data transfer objects)

**Environment variables:**  
LOGGER\_URL – logger URL. Set from docker at runtime.  
JWT\_SECRET – shared secret.  
 **Watcher Project**  
Since watcher service should continuously watch at directory for the new files, the service is implemented as a console application running background service.

Components:  
WatcherService.cs: background service using FileSystemWatcher to watch at the directory.  
On creation of the file in "watched" directory, Created event is fired and SendMetadata() is invoked. It gets file info and stores in LogMetaData DTO.  
Then create JWT token and sends POST request to logger with LogMetaData as payload and JWT attached to Authorization header. Then waits for response (Response DTO) from logger. If OK moves file to procced folder, otherwise log error.  
Settings.cs:  
Watcher settings, bounded to Watcher/appsettings.json   
Program.cs:  
Entry point.  
  
**Logger Project**

ASP.NET CORE Web API project, to serve POST requests from watcher service.  
LoggerController.cs:  
On getting the request, validates the payload and JWT token and invokes LoggerService if validation succeed. LoggerService writes data to logs directory. LoggerController then returns Response DTO with status code and message.  
Settings.cs:  
Logger settings, bounded to Logger/appsettings.json.   
Program.cs:  
Entry point.  
  
**Common project:**  
Contains the DTO classes for delivering data between services.  
LogMetaData.cs|  
Response.cs