Developer Onboarding

This service unlike other internal Amazon Music service offerings is completely detached from the Brazil/Apollo suite of tooling. As such some steps are required to configure `yarn` to correctly source packages in order to retain infosec compliance.

Local dependencies

In order to develop, build and test this application, you need Node.js and Yarn. The minimum Node.js version required is 14. In case you have multiple projects using Node.js, NVM is a tool that let you have different versions in your machine.

```
nvm install 14
nvm use 14
```

Then you can install Yarn by using the following npm command:

```
npm install --global yarn
```

AWS CLI

To work on Firefly you will need the **AWS CLI V2** installed.

For detailed instructions on how to install AWS CLI follow the AWS CLI install and update instructions

IF you have AWS CLI installed and run into any issues, make sure you are running **AWS CLI V2** 2.13.16 or a more recent version.

How to verify AWS CLI installation?

You can see which CLI version you have installed with the following commands This command should show you where it was installed

```
which aws
```

This command should show you which version you have installed

```
aws --version
```

Setting up repository permissions

This repository uses Midway for permission management. Before using Midway, SSH access must be resolved. Navigate to the SSH Access section on AWS Gitlab, and configure your SSH key.

Once created you'll need to link your SSH key to your Gitlab account.

Note: If you want to validate everything is working, try cloning the firefly package to your local workspace.

```
git clone git@ssh.gitlab.aws.dev:amazonmusic/musicfirefly/amu_webapi.git
```

Authenticating for Dev Deployments + CodeArtifact

Note that we no longer support the creation of individual IAM users for dev deployments, in order to improve security

Access to this IAM role is managed via Bindles, so you only need to `mwinit -s -o` before grabbing credentials.

In order to retrieve a new accessKey and secretKey for your AWS credentials, simply run:

```
yarn authDev <AWS_PROFILE>
```

where `<AWS_PROFILE>` is something like `firefly-dev`.
It should correspond to the `AWS_PROFILE` you export when logging into CodeArtifact.
This will overwrite the values under this profile in your `~/.aws/credentials` file. The keys will be valid for one hour, so you may need to re-run when deploying later.

Example of how `~/.aws/credentials` should look post-execution:

```
[firefly-dev]
aws_access_key_id=AQCDA....
aws_secret_access_key=bcQq25....
aws_session_token=IQoJb....
```

If you encounter issues:

- You may not have set up ada before, and should follow this guide
- You may need to `mwinit` if you get an error
- You will need to provide the name for the AWS Profile after `yarn authDev` to inform which profile to set `~/.aws/credentials`

For extra credit, automate your Firefly login with a script in `~/.zshrc`

Here's an example (make sure to replace `<INSERT ALIAS>` with your Amazon alias, i.e. `ajz`):

```
# Firefly
alias firefly='cd ~/Code/Firefly/amu_webapi ; yarn authDev firefly-dev ;
AWS_PROFILE=firefly-dev aws codeartifact login --tool npm --domain amazon
--domain-owner 149122183214 --repository music-sonic --region us-west-2 ;
export AWS_PROFILE=firefly-dev ; export USER=<INSERT ALIAS> ; export
STAGE=dev ; export NODE_OPTIONS=--max_old_space_size=8172 ; cd .. ;'
```

Configure Login and store a bash Alias:

Now with the profile configured, we can log into the MusicSonic group repository via the following command:

```
AWS_PROFILE=firefly-dev aws codeartifact login --tool npm --domain amazon
--domain-owner 149122183214 --repository music-sonic --region us-west-2
```

> Note: Our CodeArtifact repository (CodeArtifact is an Amazon-internal mirror of the public NPM repository). This layer of abstraction allows us to source both open sourced packages

> and internal packages published by Amazon. It's also worth noting that packages that do not comply with Amazon's infosec-compliance standards are not accessible.

To simplify logging into CodeArtifact and not lose session stored information on terminals. It's convenient to store the login instructions within a 1 line bash alias. Below is an example for MacOS:

`~/.bashrc` or `~/.zshrc`

```
alias ff-login='AWS_PROFILE=firefly-dev aws codeartifact login --tool npm --domain amazon --domain-owner 149122183214 --repository music-sonic --region us-west-2'
```

Now whenever we need to re-authenticate we can simply type the following:

```
~/Path/To/Dir$ ff-login
```

If you see this issue while running the above command:

```
Command '['npm', 'config', 'set', '//amazon-149122183214.d.codeartifact.us-west-2.amazonaws.com/npm/music-sonic/:always-auth', 'true']' returned non-zero exit status 1.
```

This is due to a change in how npm interprets `always-auth` since npm 7.0+. This has been fixed in recent releases of aws-cli. Use `aws --version` to check your version of aws-cli. If you are on a version earlier than 2.9.23, updating aws-cli to the latest release will resolve this issue.

Configure Yarn

Yarn is the tool used in this application, while the above lines will configure the registry. There are cases where yarn fails to resolve packages.

To avoid the below error:

```
error An unexpected error occurred: "https://amazon-149122183214.d.codeartifact.us-west-2.amazonaws.com/npm/music-sonic/axios:
```

```
Unauthenticated: request did not include an Authorization header. Please
provide your credentials.".
```

add a new line to `.npmrc` below your auth token, and write the following:

```
always-auth=true
```

Configuring Commitizen

This package and all packages under the MusicFireFlyPlatform make heavy use of semantic-release for automating deployments and version management. As such one requirement for all contributers is to install Commitizen and issue commits using this command line tool.

Configure npm to use CodeArtifact

If this is the first time during your current bash session that you are trying to install something from Code Artifact using yarn, you need to login first using the alias that we created previously:

```
ff-login
```

You should get a message that looks like:

```
Successfully configured npm to use AWS CodeArtifact repository
https://amazon-149122183214.d.codeartifact.us-west-
2.amazonaws.com/npm/music-sonic/
Login expires in 12 hours at 2023-09-07 03:00:04-07:00
```

After this, you can continue using yarn normally.

Install Commitzen

```
yarn global add commitizen
```

If cz-conventional-changelog is not installed as a dependency of commitizen, run:

```
yarn global add cz-conventional-changelog
```

and then finally

```
echo '{ "path": "cz-conventional-changelog" }' > ~/.czrc
```

Issueing a commit with Commitizen:

```
git cz
```

This will provide a prompt for styling and adding metadata to a commit that is used during the release flow:

```
➜  ng-poopy  master ✗  git add .
➜  ng-poopy  master ✗  git cz

All commit message lines will be cropped at 100 characters.

? Select the type of change that you're committing: (Use arrow keys)
❯ feat:     A new feature
  fix:      A bug fix
  docs:     Documentation only changes
  style:    Changes that do not affect the meaning of the code
            (white-space, formatting, missing semi-colons, etc)
  refactor: A code change that neither fixes a bug or adds a feature
  perf:     A code change that improves performance
  test:     Adding missing tests
  chore:    Changes to the build process or auxiliary tools
            and libraries such as documentation generation
```

Configure Deploy permissions for Sandbox (firefly-music-dev)

Install your requirements using `yarn install`

> Note: It is **crucial** that you consistently use `yarn` and **NOT** `npm` for installing dependencies. We deploy with yarn, so any installations done with npm are not guaranteed to be included in the deployment.

To deploy, perform the following commands:

```
export AWS_PROFILE=firefly-dev
export USER=<dev-Alias>
export STAGE=dev
yarn deployNA
```

Where `<dev-alias>` is your internal Amazon alias.

> *Note: MacOS automatically stores a USER environment variable that matches your Amazon Alias. As such if you are developing via MacOS you can deploy using: `STAGE=dev yarn deployNA` instead. To overwrite beta you'll need to `unset` the preset USER variable.*

> *Note: It may be convenient to place these exports into a simple .sh script you can run with an alias before calling yarn deployRegion.*

Developing against a deployed Developer stack

Once the above commands are run, you'll be presented with output relating to your newly constructed stack. It will look similar to below:

```
Serverless: Stack update finished...
Service Information
service: MusicFirefly-ryderst
stage: dev
region: us-east-1
stack: MusicFirefly-ryderst
resources: 62
api keys:
  None
endpoints:
  GET - https://f4rmipb1ea.execute-api.us-east-
1.amazonaws.com/dev/robots.txt
  HEAD - https://f4rmipb1ea.execute-api.us-east-
1.amazonaws.com/dev/robots.txt
  GET - https://f4rmipb1ea.execute-api.us-east-1.amazonaws.com/dev/
  POST - https://f4rmipb1ea.execute-api.us-east-1.amazonaws.com/dev/
  ANY - https://f4rmipb1ea.execute-api.us-east-
1.amazonaws.com/dev/{proxy+}
functions:
  logExporter: MusicFirefly-ryderst-dev-logExporter
  cloudAuthTokenWorker: MusicFirefly-ryderst-cloudAuthTokenWorker
```

```
  robotsText: MusicFirefly-ryderst-robotsText
  authorizerFunc: MusicFirefly-ryderst-authorizerFunc
  graphqlCore: MusicFirefly-ryderst-graphqlCore
  graphql: MusicFirefly-ryderst-graphql
  api: MusicFirefly-ryderst-api
layers:
  None
```

For the sample above, the GraphQL endpoint would be: `https://f4rmipb1ea.execute-api.us-east-1.amazonaws.com/dev/`.

Postman Setup

Please see this PostMan setup guide to simplify LoginWithAmazon integration and allow you to test directly against your new API endpoint.

Running the tests

There are two types of test suites available: unit and integration tests.

To run the unit tests, just run:

```
yarn test
```

To run the integration tests against the default Beta endpoint, just run:

```
yarn testIntegration
```

You can also run the integration tests against your own endpoints as deployed in the previous section. In order to do so you need to create the file `endpoint.env` in the root directory with this content:

```
NA_ENDPOINT=https://REPLACE-WITH-YOUR-NA-ENDPOINT.execute-api.us-east-1.amazonaws.com/dev/
EU_ENDPOINT=https://REPLACE-WITH-YOUR-EU-ENDPOINT.execute-api.eu-west-1.amazonaws.com/dev/
```

```
FE_ENDPOINT=https://REPLACE-WITH-YOUR-FE-ENDPOINT.execute-api.us-west-2.amazonaws.com/dev/
```

And fill it with your own endpoint values as obtained when deploying your stack to NA, EU and FE. Once you have that file, when you run the integration tests again, they will use your endpoints.

Notes

If you are testing a new client integration, make sure to add your lambda role to the AAA page for MusicFireFlyService. This allows your developer stack to retrieve the token from the client. If this is not done, the token will return null.

Common Issues

Can't create resource on AWS

When accessing the console for `firefly-music-dev`, ensure that you are properly changing the Conduit Permissions drop-down from `Read Only` to `Administrator` before clicking the `Console Access` button.

Commit is blocked

When a commit is not adhering to the Commitizen flow, or Angular notation with the `feat:` style prefixes. The githook for commit-msg will throw the below error. To resolve this problem be sure to prefix the message with a valid angular type, or use `git cz` for commitizen.

```
git commit -m "Added commit-msg hook"
⧗   input: Added commit-msg hook
✖   subject may not be empty [subject-empty]
✖   type may not be empty [type-empty]

✖   found 2 problems, 0 warnings
ⓘ   Get help: https://github.com/conventional-changelog/commitlint/#what-is-commitlint
```

Testing client service package changes with amu_webapi locally.

Since the `amu_webapi` package is a separate project from client service packages, local changes of the client need to be connected to the `amu_webapi` through the use of `yarn link` and `yarn pack` in order to test locally.

Yarn Link

Yarn link will symlink a local package (client service package) to an existing project (`amu_webapi`). The client service package will be directly linked to the node_modules of the `amu_webapi` project. Changes in the client service package will reflect immediately in the node_modules of the project that is using it. It will enable your IDE (such as VSCode) to autofill with the most recent local changes.

Step 1: Yarn link in client service package

Navigate to the root of your local client service package. Link the client service package to yarn.

```
yarn link
```

The output should resemble:

```
yarn link v1.22.10
success Registered "example-client-package".
info You can now run `yarn link "example-client-package"` in the projects
where you want to use this package and it will be used instead.
```

Step 2: Yarn link in amu_webapi

Navigate to the root of the local `amu_webapi` package.

```
yarn link "example-client-package"
```

If everything is working, the output should resemble:

```
yarn link v1.22.10
success Using linked package for "example-client-package".
```

Navigate to the root of the local `amu_webapi` package and recompile all packages.

```
yarn install --force
```

You only need to do this once unless the package is cloned again to another folder. In that case you will need to reset (described below)

To reset

Navigate to the root of your local client service package.

```
yarn unlink
```

You should see this output

```
yarn unlink v1.22.10
success Unregistered "example-client-package".
info You can now run `yarn unlink "example-client-package"` in the
projects where you no longer want to use this package.
```

Navigate to the root of the local `amu_webapi` package.

```
yarn unlink "example-client-package"
```

If everything worked, you should see this output

```
yarn unlink v1.22.10
success Removed linked package "example-client-package".
info You will need to run `yarn install --force` to re-install the package
that was linked.
```

Recompile all packages.

```
yarn install --force
```

Yarn Pack

Yarn Pack will build the client package and contain it in a compressed file which can be referenced by another project. This will allow `amu_webapi` dev deployment to include the local client changes.

Step 1: Pack

Make sure to install and build before you pack

```
yarn install --force && yarn build
```

Navigate to the root of your local client package.

```
yarn pack
```

If it ran successfully, you should be able to see a new `.tgz` file in the project folder. You should be able to see the newly created file by running `git status` in the client package.

```
example-client-package-v1.0.0.tgz
```

Run this step before each dev deployment. **Note**: If the deployment is not reflecting the changes, you may need to bump the version number of the client package. Make this change in the `package.json` in the client package under `version`.

Step 2: Point amu_webapi to the packed file

Edit `package.json` of the `amu_webapi` project. Add this line, replacing the version.

```
  "dependencies": {
    ...
    example-client-package: "file:RELATIVE_PATH_TO_CLIENT_PACKAGE/example-
client-package-v1.0.0.tgz",
```

```
  ...
}
```

The packed file is now linked and you should be able to deploy to your local dev. You only have to do this step once. **When you are ready to commit your changes,** make sure** to return this dependency to the original state.

```
"dependencies": {
  ...
  example-client-package: "^1.0.0",
  ...
}
```

Please note:

- If you encounter an error when running `yarn install`, you may have a space after `file:` that needs to be removed.
- You also might need to run `yarn cache clean`, or increment the dependency's version number, to get the yarn cache to recognize newly built changes in the `.tgz` file.

Making changes on firefly

Flow overview

- Create a local branch with your changes (ex `git checkout -b my-new-branch`)
- Make your changes and stage them (ex `git add .`)
- Add a commit using commitzen and follow the prompts (ex `git cz`)
- Push your changes to your remote branch and gitlab will provide you with a link to create a merge request (ex `git push origin my-new-branch`)

For more details on how to Contribute to the firefly codebase and our process visit the Firefly contribution guidelines

To learn more about Gitlab merge requests visit the Creating merge requests guide