Prerequisites

The easiest way to get all dependencies is to use the provided development shell in `flake.nix`. For that you will need Nix. Additionally you can use a tool like `direnv` to automatically load the environment for this repository.

The shell will setup:

- The rust nightly toolchain (nightly is required currently for ebpf because of the unstable `build-std` feature)
- The `bpf-linker`
- The Android SDK and NDK
- The `cargo-ndk` package for compiling for rust for android
- The `protobuf` programs for generating grpc server and client code

Build & Deploy

Emulator Setup

As we need a modified version of Android, we cannot use the standard system images that come with the default Android SDK. To make development easier, a custom Android SDK is loaded into your environment using the nix development shell.

> Note
>
> The Android SDK shipped with the shell is built of the standard parts with everything necessary for building Android Apps. The only "custom' part is the system image, which was build externally and is currently hosted on S3. The image can be also downloaded manually from sdk-repo-linux-system-images.zip with its manifest stored in package.xml. You can also download and unzip the system image to `$ANDROID_SDK_ROOT/system-images/android-VanillaIceCream/android-automotive/x86_64` manually and copy the `package.xml` to that directory as well.

This sdk includes the automotive system image, which is built with a custom kernel having `CONFIG_FTRACE_SYSCALLS=y` set. To create an emulator using that image, you can use the `avdmanager` tool also provided with the SDK:

```
avdmanager create avd -n YOUR_AVD_NAME  -k 'system-images;android-
VanillaIceCream;android-automotive;x86_64'  --device
```

```
vanillaIceCream;android-automotive;x86_64" --device
automotive_1080p_landscape
```

This can be started with the `emulator` tool like this:

```
emulator @YOUR_AVD_NAME
```

Frontend

The simplest way to build and test everything is the following command inside of `frontend/` (this will take a while):

```
./gradlew build
```

The apks for each build configuration are located in `frontend/app/build/outputs/apk/` .

To install the application using the real backend on your device or emulator, run either `./gradlew installRealDebug` or `adb install path/to/real/app.apk`

There are other flavors available, for example `installMockDebug` , for a frontend using fake data instead of the real backend. This is mainly interesting for development purposes.

In order to view the full list of tasks configured in gradle, run

```
./gradlew tasks
```

Backend

All in one deploy

If you'd like to build and run the daemon all in one command, you can use

```
cargo xtask daemon
```

This will ask you for root privileges to run the built executable.

By running

```
cargo xtask daemon --android
```

the executable won't start on your device but instead on an adb reachable android device or emulator by pushing it to `/data/local/tmp/backend-daemon` on the device and running it with root there.

Do it youself deploy

To just build the daemon, run the following command inside of `rust/` for your desired architecture:

```
AYA_BUILD_EBPF=true cargo ndk -t x86_64 build --package backend-daemon
AYA_BUILD_EBPF=true cargo ndk -t arm64-v8a build --package backend-daemon
```

You can then proceed to copy the executable (`rust/target/debug/backend-daemon`) to wherever you like and run it. You need root privileges in order to run it.