Environment variable configuration

Summary

Issue

We want our applications to be configurable beyond artifacts/binaries/source, such that one build can behave differently depending on its deployment environment.

- To accomplish this, we want to use environment variable configuration.
- We want to manage the configuration by using files that we can version control.
- We want to provide some developer experience ergonomics, such as knowing what can be configured and any relevant defaults.

Decision

Decided on .env files with related default file and schema file.

Status

Decided. Open to considering to new capabilities as they come up.

Details

Assumptions

We favor separating the application code and environment code. We assume the app needs to work differently in different environments, such as in a development environment, test environment, demo environment, production environment, etc.

We favor the industry practice of "12 factor app" and even more the related practice of "15 factor app".

Many of our previous projects have used the convention of a `.env` file or similar `.env` directory. There's a typical practice of keeping these out of version control, and instead using some other way to deploy them, version them, and manage them.

Constraints

We want to keep secrets out of our source code management (SCM) version control system (VCS).

We want to aim for compatibility with popular software frameworks and libraries. For example, Node has a module "dotenv" for reading environment variable configuration.

Positions

We considered a few approaches:

- Store config in the app such as in a file `config.js` file.

- Store config in the environment such as in a file `.env`.
- Fetch config from a known location such as a license server.

Argument

We selected the approach of a file .env because:

- It is popular including among experts.
- It follows the pattern of `.env` files which our teams have successfully used many times on many projects.
- It is simple. Notably, We are fine for now with the significant trade-offs that we see, such as a lack of audit capabilities as compared to an approach of a license server.

Implications

We need to figure out a way to separate environment variable configuration that is public from any secrets management.

Related

Related decisions

We expect all our applications to use this approach.

We will plan to upgrade any of our applications that use a less-capable approach, such as hardcoding in a binary or in source code.

We will keep as-is any of our applications that use a more-capable approach, such as a licensing server.

Related requirements

We will add devops capabilities for the files, including hooks, tests, and continuous integration.

We need to train all developer teammates on this decision.

Related artifacts

Each area where we deploy will need its own .env file and related files.

Related principles

Easily reversible.

Notes

Example file `.env`:

```
NAME=Alice Anderson
EMAIL=alice@example.com
```

Example file `.env.defaults`:

```
NAME=Joe Doe
EMAIL=joe@example.com
```

Example file `.env.schema` with just the keys:

```
NAME
EMAIL
```