Project Documentation / Gofore / last modified: 25.6.2024

Introduction

Virtual Finland (https://valtioneuvosto.fi/hanke?tunnus=UM012:00/2021) is a project headed by the Ministry for Foreign Affairs with aim to make it easier for foreign employees, companies, and students to relocate to Finland. As part of that goal the project produced two products that have been published in a production-like setting: Access Finland MVP and Esco API. This documentation gives a short brief of the technical side of the applications. Documentation is created by Gofore.

Access Finland MVP is an example application that utilizes the data economy idea that information could be accessible between separate organizations / services. The gist of the said utilization is in app integrations to the external service: Data Finland (datafinland.com). The application is hosted at https://accessfinland.com/. Public access to the application is restricted.

Esco API is a separate product that is also integrated to the Data Finland. It provides an easy access to the ESCO-classification datasets governed by the European Commission (esco.ec.europa.eu).

Documentation locations

Relevant GitHub repositories, documentation included:

- Virtual Finland / Projects: meta repository for Virtual Finland project deployments
- Virtual Finland / Infrastructure: common resources for the Virtual Finland project
- Virtual Finland / Monitoring: monitoring tools
- Virtual Finland / Esco API: API for the ESCO datasets
- Virtual Finland / Users API: API application for user data of the project
- Virtual Finland / Codesets: a shared codesets & resources router for the project
- Virtual Finland / Access Finland: monorepo that houses GUI web-applications for the project
  - Access Finland MVP: the relevant GUI app deployed

For developers

The components of the project are written mainly in C# and TypeScript. The bigger notable frameworks used are .Net Core/ASP.NET (Users API) and Next.js (Access Finland MVP).

The components can be run locally independently of each other for development, and each component is set up to be develop in Docker environment. VFD-tools can be used to set up and develop the project (or some parts of it) in dockerized environment (with custom local domain names), but this is not a requirement.

Some of the components might* require live AWS cloud resources for the local development to function properly. For the AWS SDK related features to work locally, the dockerized containers use the `AWS_PROFILE` environment variable with a read-only bind-mount to the local AWS configuration folder at `~/.aws`. The used AWS profile for the local development should be separate from the production.

**for example when using Sinuna.fi-service for authentication, where the backend secrets are stored in the AWS Systems Manager Parameter Store.*

The IaC-tool system (Infrastructure as Code) Pulumi is used to create, deploy, and manage infrastructure on AWS cloud. The Pulumi programs are written in C# and TypeScript.

Infrastructure

Overview

The Access Finland MVP -application and it's companion solutions are set up to Amazon AWS cloud with IaC-tool Pulumi. The different application projects and their relations are described here: Virtual Finland / Projects: deployment overview and starting point

The project is deployed to two separate and isolated runtime environments:

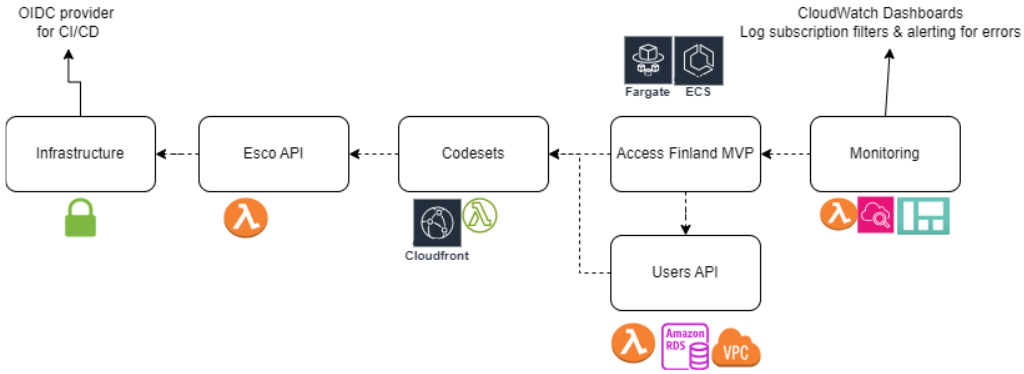
- **mvp-staging**: live environment for quality assurance
- **mvp-production**: live environment for production usage

Both environments are deployed to the AWS region `eu-north-1`, with the exception of the AWS global resources CloudFront CDN and Lambda@Edge, which have the control lane located at `us-east-1`.

Architecture

The project is structured in a microservices architecture with responsibilities separated to different projects / services.

Architecture diagram



Overview of Access Finland MVP service and it's relations

CI/CD and Pipelines

Applications in the project are configured with Github Actions pipeline that builds, tests and deploys the components.

Infra deployment

The primary dependency for all the apps infrastructure deployments is the infrastructure component. It sets up the shared resources like AWS SES (Simple Email Service) and establishes the OIDC-provider for the CI/CD-pipeline (so the applications can be deployed from GitHub Actions).

The other, application specific, infrastructure is deployed by the given apps deployment process using IaC-tooling found in the applications repository.

For insight, here's some notes of the pre-required configurations and deployment ordering when starting from scratch:

Virtual Finland / Projects: setup from scratch

Application deployment

The apps can be deployed (to a specific runtime environment) individually from the applications repository actions (for example: Deploy AF MVP · Workflow runs · Virtual-Finland-Development/access-finland (github.com)) or with a project-wide deployment action at the projects-repository actions tab: Deploy MVP Phase 2 · Workflow runs · Virtual-Finland-Development/projects (github.com).

Each application can also be deployed manually with command-line tools and should have deployment instructions in the repository readme-files.

The deployments are triggered manually.

Application(s)

Access Finland MVP

The projects primary application that provides the web UI.

Access Finland MVP is a Next.js application that runs as a Fargate task in the ECS (Elastic Container Service) that is accessed through CloudFront distribution and ALB (Application Load Balancer). The public access to the application is restricted using a combination of WAF (Web Application Firewall) and Cognito. Additionally the application implements internal authentication using a separate Cognito user pool.

The stack:

- Turborepo & Next.js
- ECS / Fargate
- ALB
- CloudFront:
  - is configured to be used with runtime specific custom domain names:
    - **mvp-staging**: https://staging.accessfinland.dev
    - **mvp-production**: https://accessfinland.com
- WAF
- Cognito: `wafUserPool`
  - manages public access to the site
- Cognito: `loginSystemsUserPool`
  - manages authentication for the site users

The documentation pages of the Access Finland MVP:

- General
- Deployment
- In-app authentication
- Security

The Access Finland code repository is structured as a monorepository that contains multiple apps. From there, the app of interest is referenced as `af-mvp`.

Users API

Component that handles the data persistence and some of the application logic. Has access to the databases.

Users API is an ASP.NET / Entity framework application that is deployed as an AWS Lambda function URL. The app utilizes a Postgresql-database for data persistence and a Redis-database for caching and is deployed in an isolated VPC (Virtual Private Cloud). The application also has some management tasks implemented as separate lambda functions.

The stack:

- Entity Framework
- AWS Lambda Function(s)
- Lambda Function URL
- CloudWatch EventBridge
- SQS
- VPC

- RDS Aurora Postgresql
- ElastiCache Redis

The documentation pages of the users-api can be found here: users-api/Docs at main · Virtual-Finland-Development/users-api (github.com)

Codesets

Serves static code sets, like municipality codes, to the applications.

Codesets is a Cloudfront / Lambda@Edge application written in Typescript that use S3 buckets as internal cache (and cloudfront as external).

The stack:

- Node.js / Typescript
- CloudFront
  - frontend / edge-cache
- Lambda@Edge
  - resource resolver
- S3
  - cache for the external resources
- Lambda Function:
  - function that is used to re-generate caches on post-deployment phase

The documentation / repository: Virtual-Finland-Development/codesets: A shared codesets / resources router for virtual finland projects (github.com)

Esco API

Serves the static ESCO codes to the applications. Separated from the Codesets-service as independent service because it's used by third-party operations.

Esco API is a Bun.sh / Typescript application that runs as AWS Lambda function URL endpoint.

The stack:

- Bun.sh / Typescript
- Lambda Function
  - custom runtime: `provided.al2`
- Lambda Function URL

The documentation / repository: Virtual-Finland-Development/esco-api: API for the ESCO datasets (github.com)

Databases

Profile database

The profile database is an AWS RDS (serverless v2) cluster that is managed by the #3.4.2-Users-API deployment flow.

Database has the following properties:

- a name like `users-api-postgres-db`.
- users:
  - `apiDbUser`: API-application user with restricted grants
  - `dbAdminUser`: administrative user with admin grants, used with admin functions of the users-api.

Cache database

The cache database is AWS ElastiCache redis cluster that is managed by the #3.4.2-Users-API deployment flow with a name like `users-api-rediselasticache`.

Integrations to other platforms

Dataspace API Gateway

The project is integrated to the dataspace -API gateway.

The live environments use the following dataspaces:

- **mvp-staging**: Dataspace Portal | Data Finland Staging | Data Finland Staging
- **mvp-production**: Dataspace Portal | Data Finland

The integration shortly described with an example:

- Access Finland MVP sends data requests with agreed payload schema to the Dataspace
- Dataspace forwards the request to external endpoint (for example our own provided Users API).
- External endpoint responds with data formatted to agreed response schema
- Dataspace validates the response and responds to the original request

→ The Dataspace acts as a guard of the data schema agreement.

The integration implementation source code is located here: access-finland/apps/af-mvp/src/pages/api/dataspace/[[...slug]].route.ts at main · Virtual-Finland-Development/access-finland (github.com)

Job Market Finland public API -endpoint

Access Finland MVP app utilizes the Job Market Finland public API recommendations-endpoint in retrieving the skill-, and occupation related classifications to the job searching profile. Unfortunately the API is not documented publicly.

The integration implementation source code is located here: access-finland/apps/af-mvp/src/pages/api/jmf/recommendations.route.ts at main · Virtual-Finland-Development/access-finland (github.com)

The API endpoint address: `https://tyomarkkinatori.fi/hakupalvelu/api/1.0/skillrecommendation/skillrecommendation/`