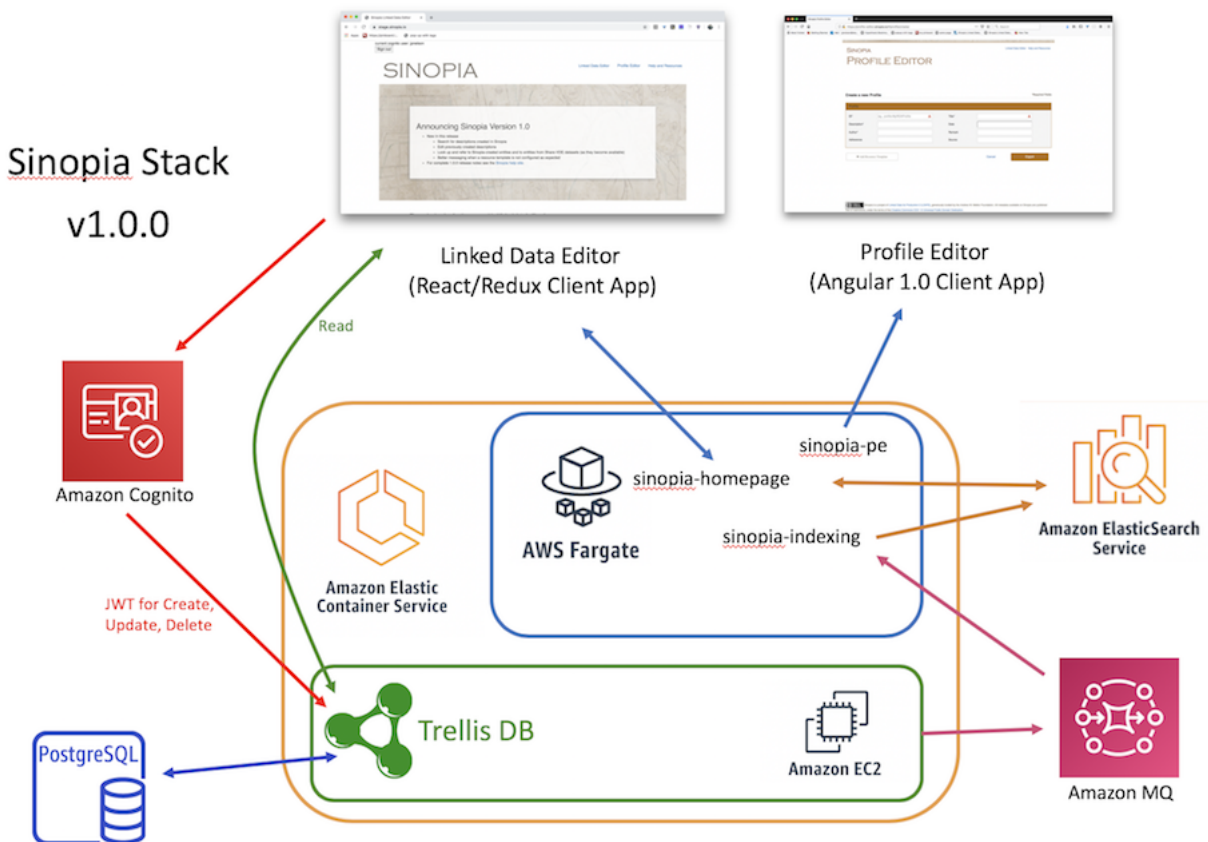





Running the Sinopia Stack on Amazon Web Services



Background

An early requirement for the open-source [Sinopia](#) project was to build a cloud-based collaborative editing environment. The team chose [Amazon Web Services](#), one of the most popular commercial cloud providers, by utilizing a number of different AWS services and products to host Sinopia and its dependent technologies. Although the first version of Sinopia is closely tied to specific AWS cloud services, we are looking at more generic infrastructure options for hosting Sinopia on other commercial cloud services as well as in hybrid environments.

Trellis - Linked Data Platform



In Sinopia's early analysis, we determined that having a RDF triplestore was not necessary for meeting the requirements of a create-update-read-delete (CRUD) editor for RDF. Looking at the available options, we realized the [linked-data platform Trellis](#) was a good match to our requirements.

An open-source project started by Aaron Coburn, [Trellis](#) offers multiple storage backends, including a relational database, that stores entity or named graphs. Trellis also provides a message queue that we could leverage for indexing these named graphs into an AWS Elasticsearch.



Amazon Web Services Cognito

To handle authentication and authorization for create, update, and delete operations on the Linked Data RDF and JSON resource templates in Sinopia, we

are using the AWS [Cognito](#) secure user sign-up and access control service.

After a user successfully signs-up via the [Amplify](#) SDK, a [JSON Web Token](#) is generated that allows authorized access to the Sinopia Linked Data Editor [React](#) components. The JWT is also used for write and edit HTTP actions for both JSON and RDF payloads that are managed in [Trellis](#), Sinopia's Linked Data Platform.

[Top](#)

Amazon Web Services

Elasticsearch Service



For the initial 1.0 release, a simple search index was created that indexes a small subset of the RDF created in editor through the AWS hosted [Elasticsearch Service](#). Elasticsearch is a full-text search engine, based on [Lucene](#), that has been optimized for running on Amazon's cloud.

Amazon Web Services

Elastic Container Service

The deployment of Sinopia on AWS relies on pre-built Docker images hosted on [DockerHub](#) that are then run in a [Elastic Container Service](#) (ECS) cluster. Sinopia is run on three [ECS](#) clusters; development, staging, and production with corresponding Docker images for each environment.



Amazon Elastic Container Service

[Top](#)



Amazon Web Services Apache ActiveMQ

Trellis publishes events like creating, updating, or deleting resource as they occur to a AWS [Messaging Queue](#) that is monitored by a Docker container `process sinopia-indexing` pipeline that then updates the [Elasticsearch](#) search index.

[Apache ActiveMQ](#) is Java-based open-source message broker that connects multiple servers and clients. For Sinopia, Trellis publishes messages for create, update, or delete actions for RDF to the queue and our indexing pipeline [Docker](#) container responds to these messages by updating our [Elasticsearch](#) index based on the action in [Trellis](#).

[Top](#)

Amazon Web Services

Fargate



[Fargate](#) is an AWS service that runs [Docker](#) containers in a state-less fashion within a Virtual Elastic cluster. A [Docker](#) container is a lightweight isolated Linux executable package of software that is generated in a deterministic way from a [Docker](#) image.

In Sinopia, we run the following Docker Images as [Fargate](#) tasks:

- Sinopia Linked Data Editor [Docker image](#)
- Sinopia Profile Editor [Docker image](#)
- Sinopia Indexing Pipeline [Docker image](#)
- Sinopia ACL [Docker image](#)

[Top](#)

Amazon Web Services

Relational Database Service (RDS)



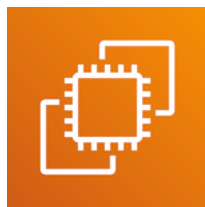
Sinopia server's core technology is a variant of [Trellis](#) Linked Data Platform that uses a [PostgreSQL](#) relational database to manage the Sinopia Editor's RDF and JSON payloads instead of a RDF Triplestore.

The AWS RDS services offers the following advantages:

- Significantly cheaper than AWS Triplestore [Neptune](#)
- Automatic backups
- Reduced administrative overhead

[Top](#)

Amazon EC2



The other method we use to run a [Docker](#) image is by hosting as part of a virtual machine running as a [EC2](#) Docker container. When we started Sinopia, [Fargate](#) does not persist data so we could not mount a permanent disk volume to store the RDF entities [Memento](#) metadata as needed by our Linked Data Platform, [Trellis](#).

[Top](#)

Infrastructure Future(s)

Although [Sinopia](#)'s infrastructure is container-based, built using [Docker](#), much of the implementation choices were done in the most expedient manner by trying to use Amazon Web Services in as many places in the infrastructure as possible. We recognize the danger of being too tightly coupled with a particular cloud provider and in a future work-cycle (we're always open to code Pull Requests!) we may look at using a more open, cloud-neutral tool chains and approaches.

Using Trellis Triplestore

During Sinopia's initial requirements gathering for the core functionality required for a minimal viable product, a RDF Triplestore was not necessary for managing the RDF being generated from the editor. Future requirements may require a Sinopia RDF triplestore, especially looking at supporting something like [SHACL](#). Fortunately, [Trellis](#) supports both a Triplestore and relational database backends.

Docker Swarm and/or Kubernetes

Using Amazon Web Service [Fargate](#) for the initially MVP was acceptable for running specifically on Amazon's platform. For the long-term, Sinopia should be able to run in any [Docker](#)-based infrastructure like [Docker Swarm](#), or more likely, run under a [Kubernetes](#) cluster for container orchestration and management.

[Questions?](#)

Thank-you!