Build Scientific Computing Infrastructure with Rebar3 and Docker

Eric Sage

A scientific telecommunications network

“Hello, I’d like an automated gene ontology please!”

Agenda

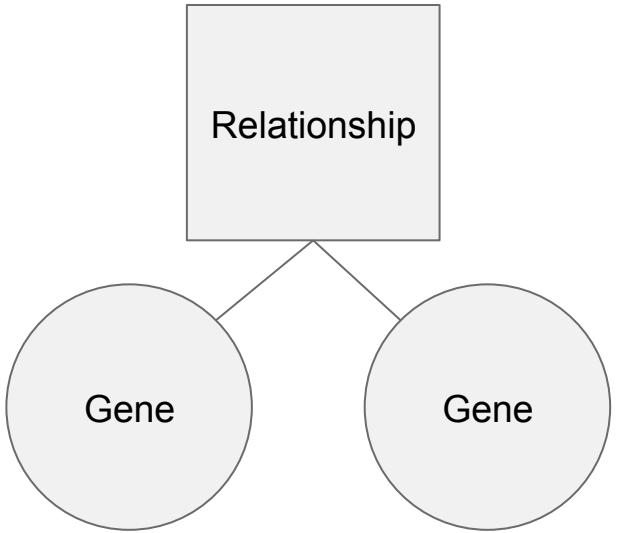
- An example biological service
- Gather requirements
- Bad Solutions
- Our Solution
- Details
- Evaluation

Biology today, a quick example:

Gene Annotations

A network where leaves represent genes and parents represent the relationships of genes (Gene products).

Example: GO (Gene Ontology) a way of standardizing the annotation.



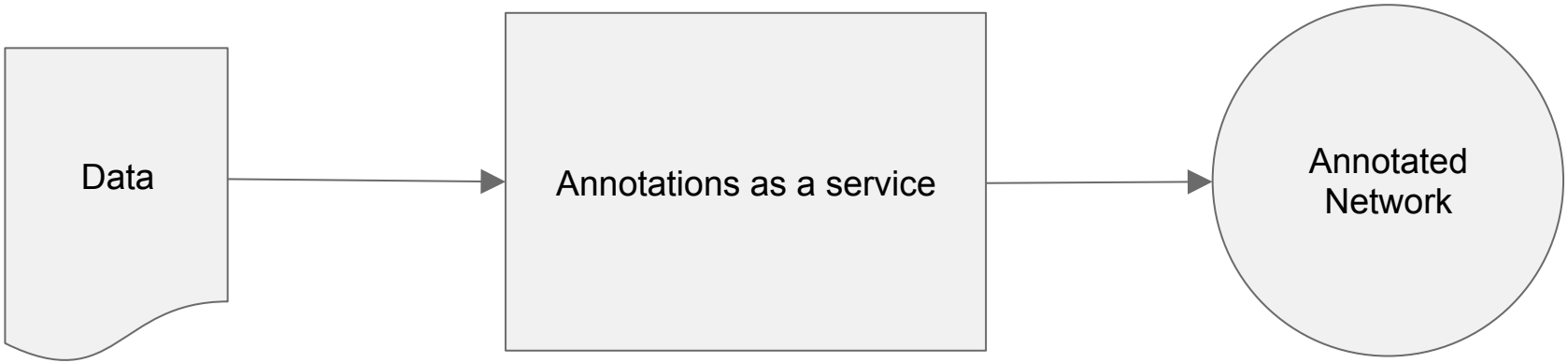
What can we do with a biological network?

- Differential Analysis - analyze and predict perturbations and alterations
- Big data integrations - models for clinical diagnoses and predictions
- multi-scale - link networks together to create hierarchical

Going beyond GO: Automation

Gene annotations as a service

Example: atgO, builds annotations using machine learning



Annotation as a service

The Goal

A world where a biologist can write biologically useful code that can last and make it available to the masses.

A Problem with the Computational Ecosystem

Biologists:

- Have little time to invest
- Construct poor implementations
- Suffer from reinvention

Software is hard, Biology is hard, you should only pick one.

Gathering Requirements

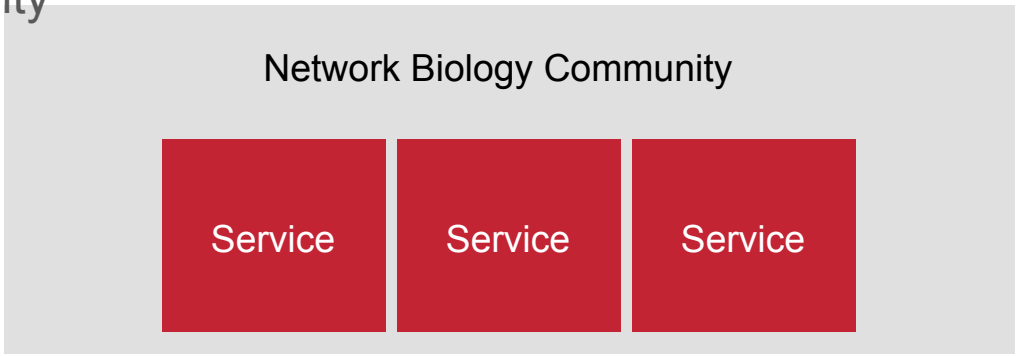
We need systems that we can

- scale
- distribute
- locate
- evolve
- verify

A Service Oriented Architecture Approach

With a SOA we can provide

- Standardization
- Discoverability
- Evolvability

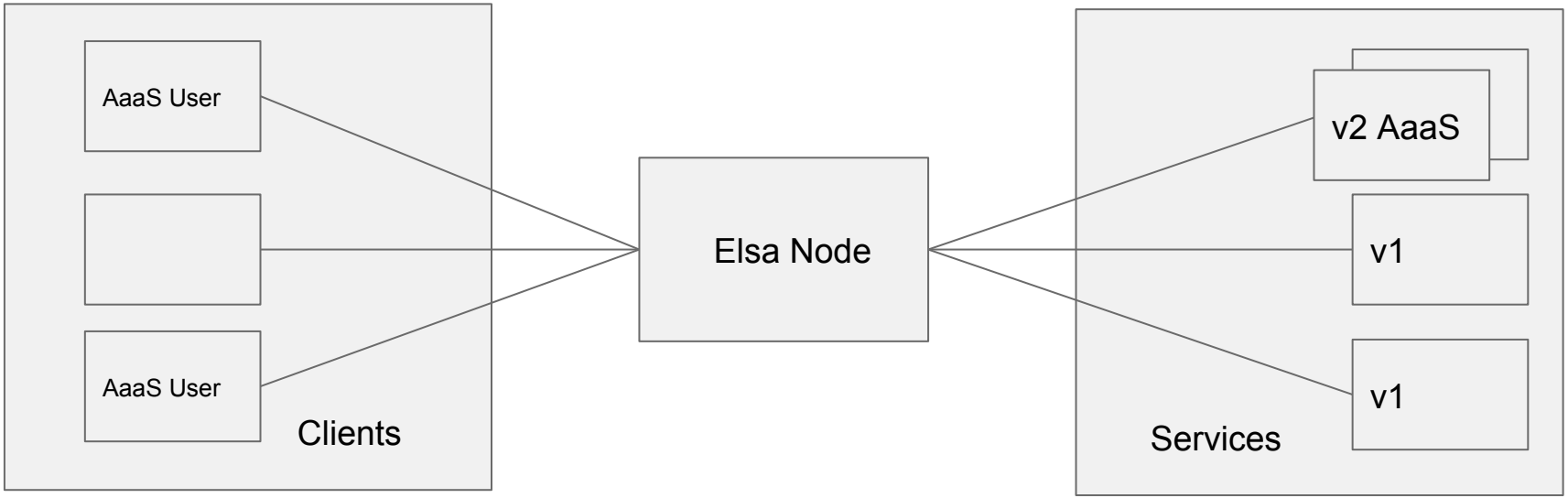


How to implement a SOA? A few OK ideas

- Enforce an ecosystem
- Enforce quality (The Jeff Bezos mandate)
- Enforce an enterprise tool (Enterprise Service Bus)

We need to shepard emerging computational communities with as little overhead as possible.

Elsa: The Erlang Submit Agent

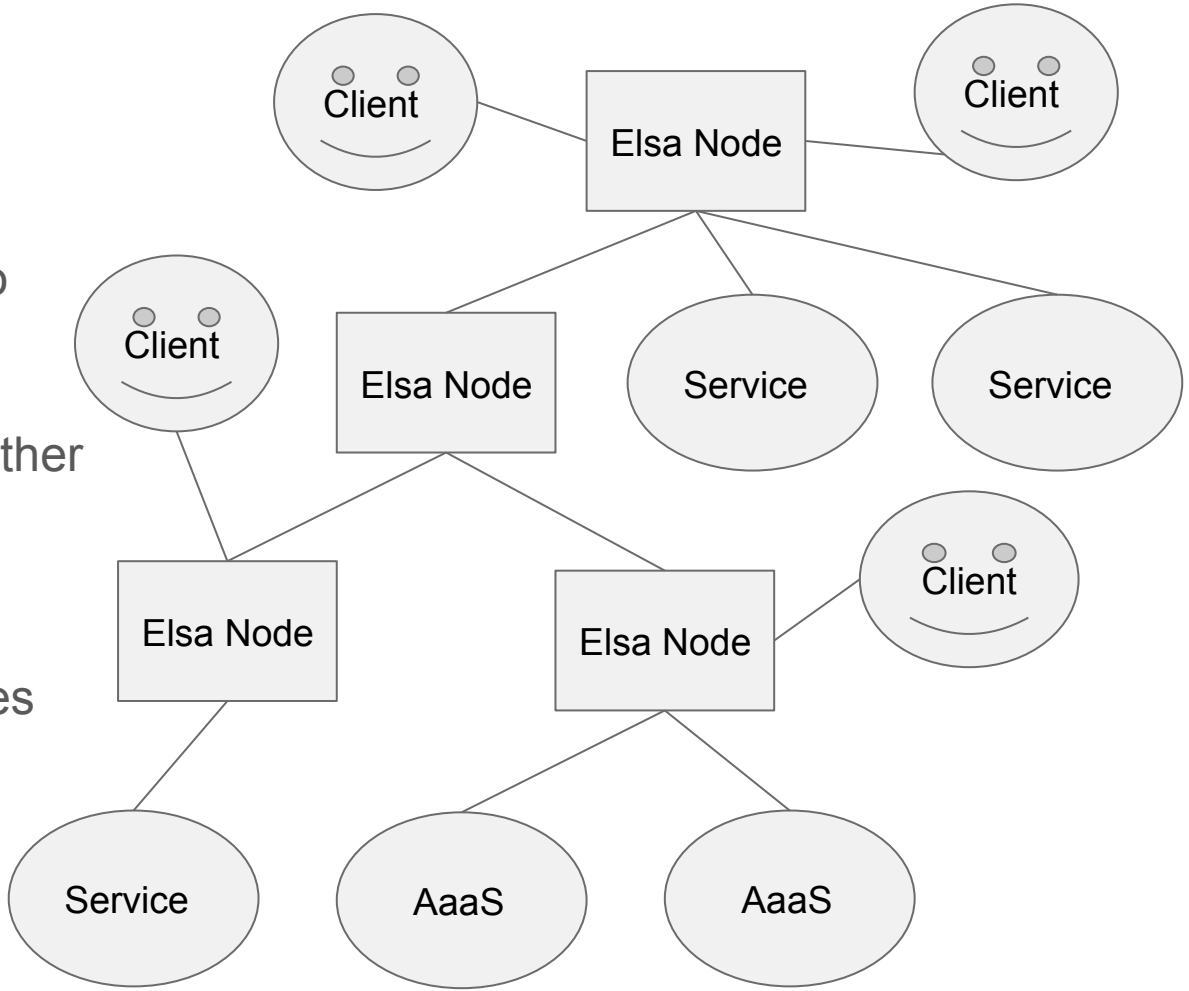


- Allows clients to submit jobs to a network of services.
- Provides a protocol for long running jobs, a must have for scientific computing.
- Handles service discovery, service versioning, and connection quality

Service Domains

Elsa as a Relay network

- Clients send requests to nodes.
- A node can relay to another node in order to fulfill a request.
- A set of connected nodes
- forms a domain.



Service Domain API

Elsa also provides a restful API available to clients, services, and external tools that provides:

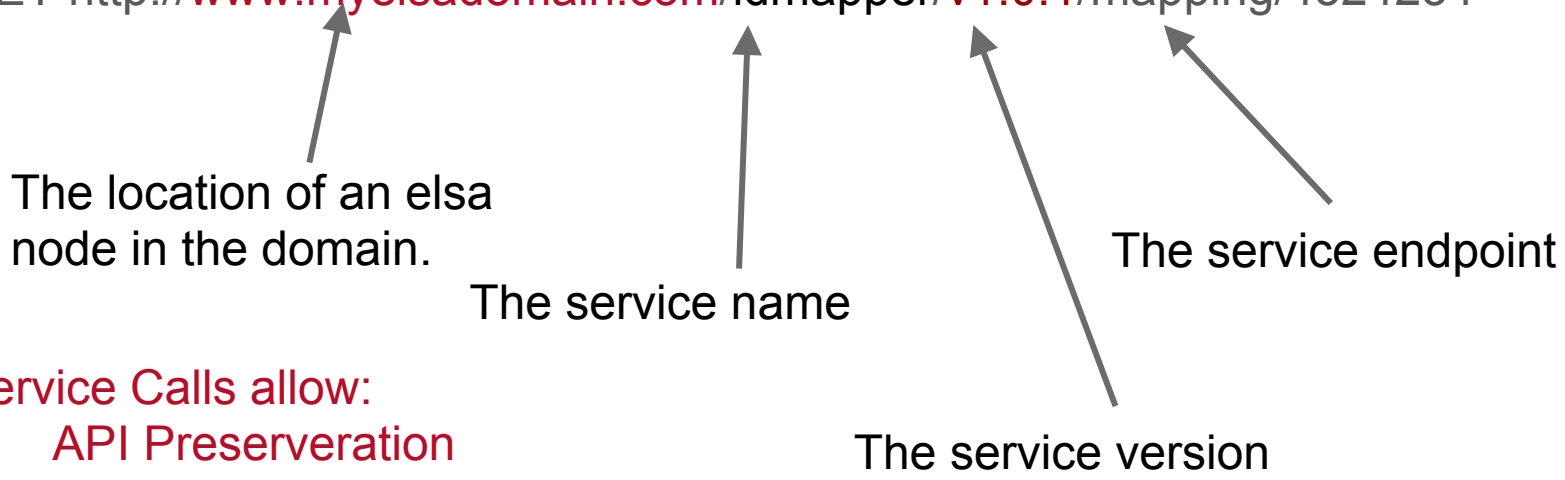
- Service Discovery
- Task Resources
- Introspection
    - Atomic resources (threads)
    - Logs

A regular service call

GET http://www.mappingservice.com/mapping/4324234

The anatomy of a service call

GET http://www.myelsadomain.com/idmapper/v1.0.1/mapping/4324234

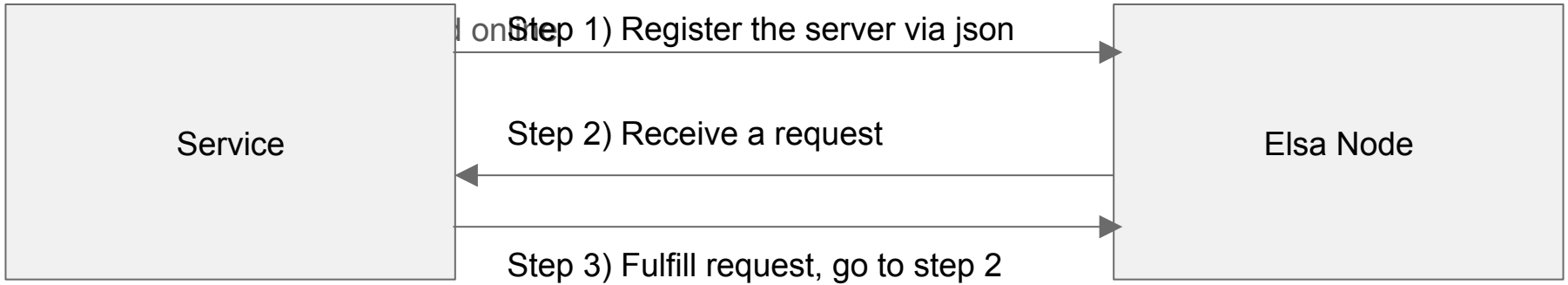


Service Calls allow:

- API Preserveration
- Advanced versioning
  - v0.1.0^
  - v0.1.*

If you know REST, you know how to build a service

- All scientists create REST services.
  - Vast majority of languages provide easy to use web frameworks
  - A short jump to convert a CLI command or any function to a REST handler



A sample registration

```
{
  service: “idmapper”,
  version: “v1”,
  instances: [ We can register as many instances as want
    {
      location: “http://123.321.123.321:8080”,
      threads: 32, Optional
      syslog: “http://syslog.myservices.org:3000” Optional
    },
    {
      location: “http://123.321.123.321:8081”
    }
  ]
}
```

Elsa handles

- Versioning
- Finding instances
- Forwarding pertinent logs
- Load balancing
- Discovery
- Long running jobs
- Partial service failure

Erlang provides robustness in the face of questionable service quality!

Deploying Nodes without fear

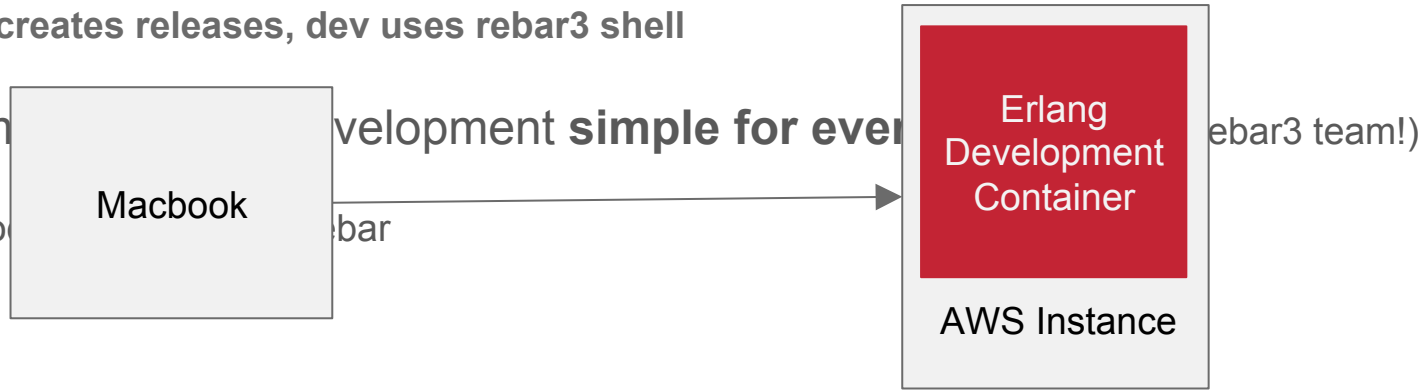
Even if it’s easy to call a service, and write a service, deploying nodes must also require little effort.

Tools in the arsenal:

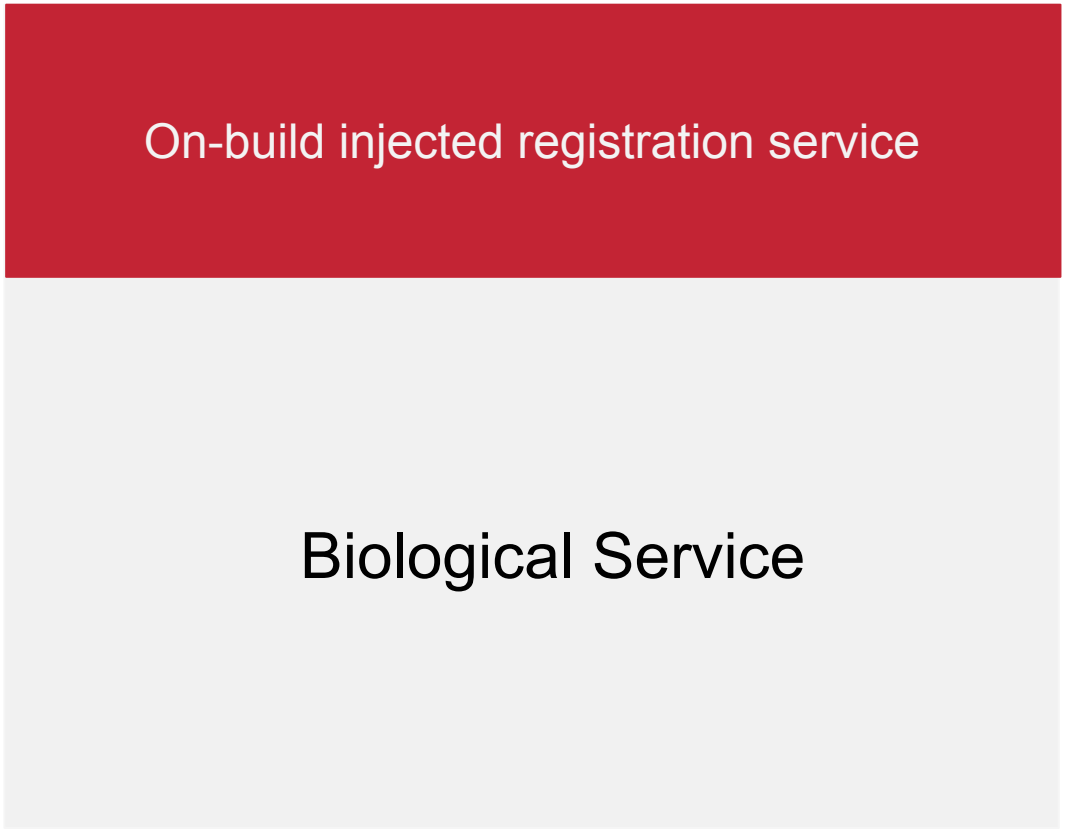
- Rebar3
- Docker
- OTP

Elsa Development

- Elsa, written from **inside** of a container
- Two Dockerfiles, dev and prod
  - Dev: Uses a large custom dev base image
  - Prod: The sys.config get's swapped for a container specific **sys.config.docker**
  - **Prod creates releases, dev uses rebar3 shell**
- Rebar3 m velopment **simple for ever** ebar3 team!)
  - Set do ebar



Service Deployment: Docker on-build



FROM bioservice:R
FROM bioservice:Python
FROM bioservice:Node

Note: This isn't the same as a per language framework

Deployment made easy

- Just use Docker
    - Deploy nodes, services using containers
    - Deploy entire domains using compose or kubernetes, pick your favorite

Note: Elsa Nodes need to run one container to a host!

Elsa Evaluation

Pros:

- **Little investment**
- **Easy to use**
- Benefits of SOA
- Central point of development

Con:

- MS delay for round trip requests
- Central point of failure

Examples of computing in biology

1. Protein structure determination with X-ray crystallography or NMR.
2. Whole genome sequencing and assembly
3. Simulation of biomolecules with molecular dynamics
4. Gene expression analysis
5. Phylogenetic (evolutionary) analysis

Synthetic Evolution: Where to go next

- Administration GUI
- Pluggable service scheduling
- Optional service patterns
- Reduce node latency
- Increase distributability
- Global Elsa network

Acknowledgements

- Dr. Barry Demchak and Dr. Trey Ideker
- Cytoscape team and Ideker lab
- UCSD Health Sciences

Going Further

- http://www.cytoscape.org/ Cytoscape Website
- https://www.github.com/cytoscape-ci/elsa/ Project Elsa
- https://www.github.com/ericsage/neoelsa/ Next Version of Elsa