Systems engineering challenge

Local development environment:

I moved the development environment to use Docker-compose for ease of use, I think it's a nicer way of getting a developer up and running and also controlling dependencies within services.

Setup steps:

I wanted a nice way of creating the databases with scripts instead of doing it by hand so I created a shell script (ticker-pg container must be running). I would much prefer importing in the Active Record tools to do this normally. I wasn't a big fan of how the db setup.rb script was written so I changed it only to check on production and the rest just uses the Docker container for testing and development.

- docker-compose up ticker-pg
- scripts/create_databases.sh
- scripts/run_migrations.sh
- scripts/run_tests.sh

Deploying the application

The underlying AWS infrastructure was built with Terraform. This creates and manages the VPC, security groups, internal/external DNS records with Route53, load balancer and auto scaling group. Ideally this would include tests and run from a centralised location (CI server) for production after passing its tests.

I would generally use Kubernetes for container orchestration but I was interested in seeing how I can deploy containers with Fleet and Systemd (more of a POC). I've come across Fleet in building Kubernetes clusters but I haven't had much of a chance to take a deeper look at it.
In conjunction with Systemd and Cloud-config it's able to pull and run images as a service, these instances this is running on then attaches to the ELB. I would generally have a lightweight health check for ELB to check again.

I normally use the Elastic Container Registry and would have this under configuration management separate from the application, this allows for the infrastructure to torn down but we still have application Docker images. Unfortunately the problem we have now is the chicken before the egg, there won't be any images in our newly created ECR repository.

This is in no regards a ‘production’ ready system, if I was to make this production ready I would do the following

- Produce a full Redis cluster and use Redis sentinel
- Enable multi region RDS instance
- Implement logging and monitoring
- Route53 internal and external DNS records

Deploying the system

- cd terraform/ecr; terraform apply
  (This will create the Container Registry)
- Run make all in the application directory (This will produce and push a image the ecr, you will need to login)
- cd terraform/cluster; terraform apply
  This will build all the other necessary infrastructure (bar DNS records)

From there the instances should start - pull down the required Docker images and start their services.

It should be in theory then possible to see them with

fleetctl --tunnel <IP[:PORT]> list-units

I generally use a SSH proxy configuration through the Bastion so this something else that would need adding. I think that’s everything, unfortunately I have been unable to test this as I don’t have access to an AWS account currently. I’m happy to talk you through this when I come into the office though.