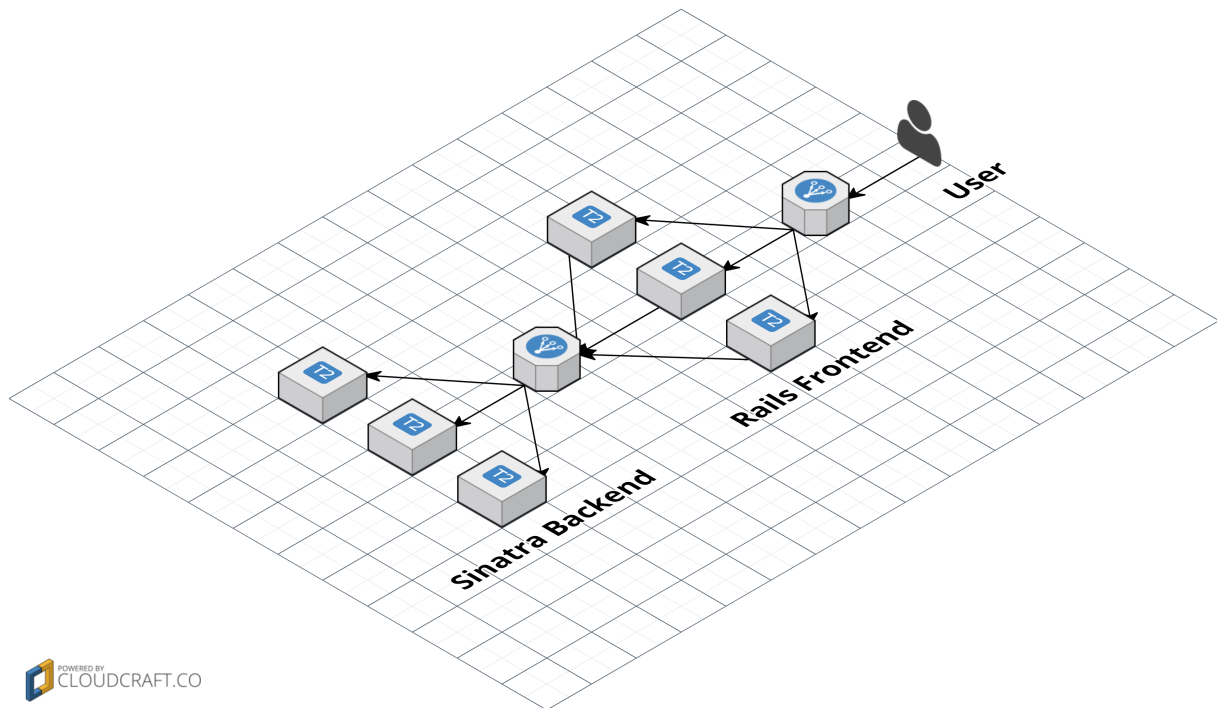


---

## Infrastructure as Code Talk

This repo contains the sample code for the talk Infrastructure-as-code: running microservices on AWS with Docker, Terraform, and ECS. It includes a couple sample Dockerized microservices and the Terraform code to deploy them on AWS:



**Note:** This repo is for demonstration purposes only and should NOT be used to run anything important. For production-ready version of this code and many other types of infrastructure, check out Gruntwork.

### Quick start

#### Running the microservices locally

To run the rails-frontend and sinatra-backend on your local dev box:

1. Install Docker.
2. `docker-compose up`
3. Test sinatra-backend by going to `http://localhost:4567`.
4. Test the rails-frontend (and its connectivity to the sinatra-backend) by going to `http://localhost:3000`.

---

The `docker-compose.yml` file mounts `rails-frontend` and `sinatra-backend` folders as volumes in each Docker image, so any changes you make to the apps on your host OS will automatically be reflected in the running Docker container. This lets you do iterative “make-a-change-and-refresh” style development.

## Deploying the microservices in AWS

To deploy the microservices to your AWS account, see the terraform-configurations README.

## Using your own Docker images

By default, `docker-compose.yml` and the terraform-configurations are using the `gruntwork/rails-frontend` and `gruntwork/sinatra-backend` Docker images. These are images I pushed to the Gruntwork Docker Hub account to make it easy for you to try this repo quickly. Obviously, in the real world, you’ll want to use your own images instead.

Follow Docker’s documentation to create your own Docker images and fill in the new image id and tag in:

1. `docker-compose.yml`: the `image` attribute for `rails_frontend` or `sinatra_backend`.
2. `terraform-configurations/terraform.tfvars`: the `rails_frontend_image` and `rails_frontend_version` or `sinatra_backend_image` and `sinatra_backend_version` variables.

## Overview of the repo

Here’s an overview of what’s in this repo:

1. An example sinatra-backend microservice that just returns the text “Hello, World”. This app includes a Dockerfile to package it as a Docker container.
2. An example rails-frontend microservice that makes an HTTP call to the sinatra-backend and renders the result as HTML. This app includes a Dockerfile to package it as a Docker container.
3. A `docker-compose.yml` file to deploy both Docker containers so you can see how the two microservices work together in the development environment. To allow the services to talk to each other, we are using Docker Links as a simple “service discovery” mechanism.

- 
4. Terraform configurations to deploy both Docker containers on Amazon's EC2 Container Service (ECS) so you can see how the two microservices work together in the production environment. To allow the services to talk to each other, we deploy an Elastic Load Balancer (ELB) in front of each service and use Terraform to pass the ELB URLs between services. We are using the same environment variables as Docker Links, so this acts as a simple “service discovery” mechanism that works in both dev and prod.

### **More info**

For more info, check out the talk [Infrastructure-as-code: running microservices on AWS with Docker, Terraform, and ECS](#), including the video and slides. For a deeper look at Terraform, check out the book *Terraform: Up & Running*.