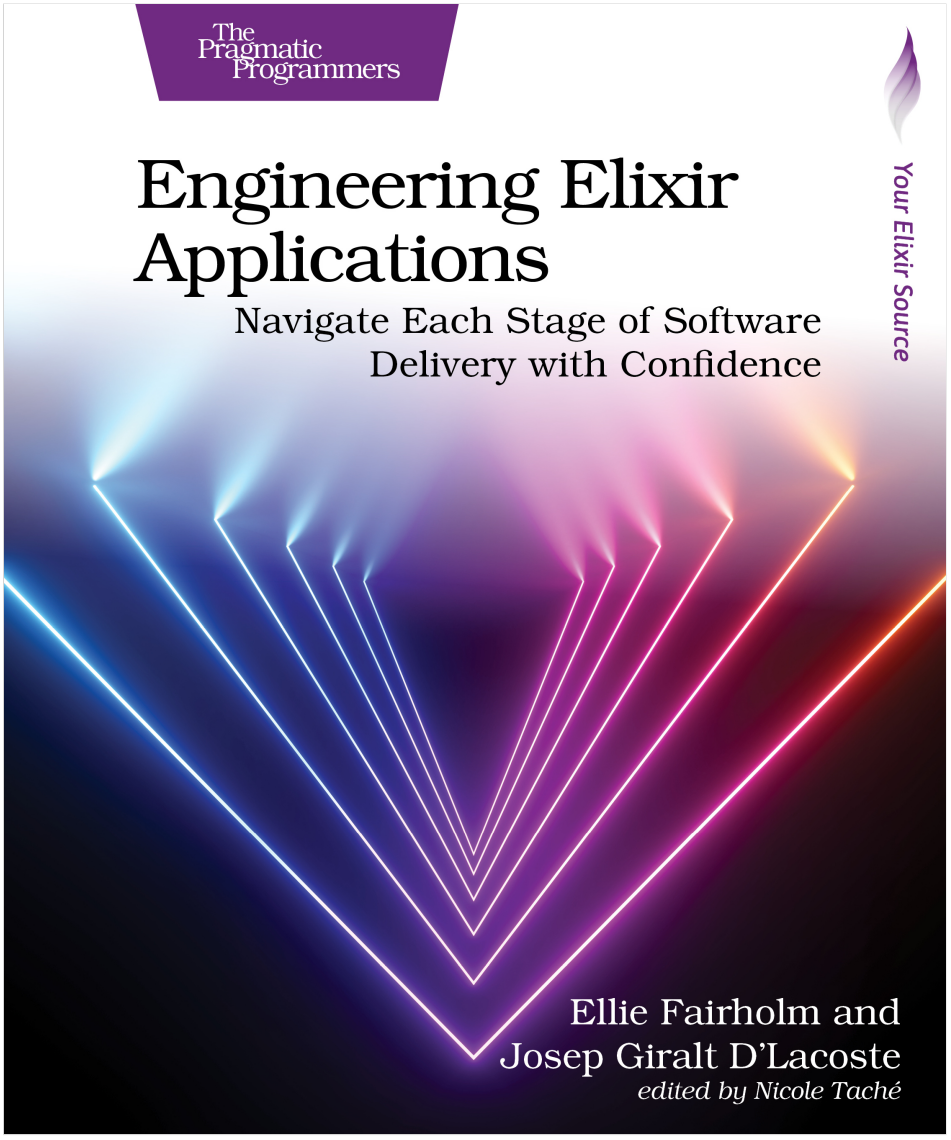

This extract shows the online version of this title, and may contain features (such as hyperlinks and colors) that are not available in the print version.

For more information, or to purchase a paperback or ebook copy, please visit https://www.pragprog.com.

Going Back to the Root of the DevOps Paradigm

As engineers or developers, working in our development environments is comfortable. Developing new features comes easily, but transitioning those features from our local machines to a production environment often seems like a mystical process that "just happens." This is due, in large part, to the separation of "dev" and "ops" people within software teams.

The "DevOps" concept was born out of the need for engineers to be able to tackle anything, whether on the dev side or the ops side. Nowadays, however, the vast majority of "DevOps" teams are really "ops" teams that aren't really aware of what's going on in the application layer, or "dev" side. We want to help change that. We want to take engineering teams back to the root of what "DevOps" is. We want to empower you to be a multi-disciplinary programmer who understands not only how to develop software but also how to deploy it. Being able to do both is a super-power in today's tech industry. It gives you confidence in what you are deploying and a more comprehensive understanding of how new features behave in a real-world environment.

To that end, this book introduces a new paradigm, BEAMOps, which is a more niche version of DevOps that focuses on developing BEAM applications. Core BEAMOps principles include environment integrity, scalability, and infrastructure as code. By pairing these principles with the fault tolerant nature of the BEAM, you will deliver reliable and robust software applications.

In this book, we'll guide you through each of the BEAMOps principles and explain how you can apply them to each step of software project delivery, from start to finish. You'll learn how to manage a project, package an application, create integration pipelines, provision infrastructure, build base machine images for remote servers, automate deployments, distribute an

application as part of an Erlang cluster, orchestrate an application, autoscale an application, and finally, instrument an application.

That's enough about the book for now. Let's talk about YOU.

Is This Book For You?

This book is written for advanced beginners or intermediate programmers who are familiar with Elixir and basic shell usage, but who feel stuck when it comes to deploying and scaling their applications. The complexity of the Elixir code is minimal; rather, the focus is on how to package and ship an application to production in a reliable and iterative manner. As a result, managers of software teams with little coding experience can also benefit from the general themes that we explore.

The techniques discussed in this book are suitable for mid-size or personal projects, as they will give you a basis for understanding all the steps involved in shipping an application. The efficient programming practices in this book will set you up with transferable skills that can be applied to many different technologies.

We've written this book for people who like to "jump over the fence" and get stuck in an issue rather than "throw it over the fence." We want to empower you to learn something new and not be scared of the unknown. After all, magic often happens when you're outside of your comfort zone.

What's In This Book?

Reading this book will feel like we're taking a journey together. Each chapter tackles a specific part of the software delivery process and builds on content from the previous chapters. For this reason, we recommend that you read it sequentially.

Chapter 1: Introduction to the Journey, on page ?

This chapter will formally introduce you to the BEAMOps paradigm and its core principles. You'll learn how the different stages of software delivery relate to the BEAMOps principles. You'll also set up your development environment and install all of the tools you'll need to complete your journey.

Chapter 2: Using Terraform to Create GitHub Issues and Milestones, on page ?

Terraform is an infrastructure-as-code tool that lets you safely provision and manage infrastructure. This chapter will discuss the project management

stage of delivering a software product. It will ease you into Terraform by demonstrating how you can use it to create familiar GitHub resources: repositories and issues. You'll create issues and milestones for each of the different stages of software delivery covered in this book. And in each of the subsequent chapters, you'll close at least one of the issues/milestones that you created.

Chapter 3: Building and Dockerizing a Phoenix Live View Application, on page ?

Docker is a platform designed to help you build and run containerized applications so that they can be shared and run in a consistent manner. It is essential for creating integrity between environments. This chapter will guide you through creating a production-ready build for a Phoenix Live View application. You'll create a basic Phoenix application and then take a crash course in creating OTP releases and containerizing your application using Docker.

Chapter 4: Setting up Integration Pipelines with GitHub Actions, on page ?

Integration pipelines are an important part of the software delivery process – they help you ensure that your code works as you expect it to. In this chapter, you'll write your own continuous integration (CI) pipeline using GitHub Actions. We'll explore both mandatory and non-mandatory steps for an ideal CI for an Elixir application.

Chapter 5: The Dev Environment with Docker Compose, on page ?

You want to ensure environment integrity for both simple and more complex applications. In this chapter, you'll create an application suite that is made up of more than one service. To do this, you'll add a database to the application that you created in Chapter 3. You'll also learn about Docker Compose, Docker Swarm, and the Docker CLI commands needed to run your application locally. This will allow you to create an environment template for a multi-service application.

Chapter 6: The Production Environment and Packer, on page ?

This chapter will cover the provisioning of remote environments. You'll be introduced to Packer, a tool that allows you to automate the creation of the base machine images for virtual machines. You'll combine Terraform and Packer to create/provision a production environment in AWS and then deploy your basic application to a single-node swarm in production.

Chapter 7: Continuous Deployment and Repository Secrets, on page ?

Continuous deployment (CD) is an essential part of the software delivery process. It ensures environment integrity by making sure that your application is always deployed to production in the same way. In this chapter, you'll create a CD pipeline with GitHub Actions that automates the deployment of your application once code is merged into the main branch of your repository. You'll also learn how to hide any sensitive data in your code using SOPS and make them available to your application with Docker secrets.

the (as yet) unwritten content

Distributing your infrastructure is important, as it makes the software you deliver more fault-tolerant. In this chapter, you'll adapt your production Terraform configuration to create a multi-node swarm, rather than a single-node one. To do this, you'll learn about AWS services such as the SSM Parameter Store and IAM roles/policies. You'll make your application highly available by distributing the different nodes in your Docker swarm in different AWS availability zones (or data centers).

the (as yet) unwritten content

Connecting multiple replicas of Elixir applications into a distributed cluster is a great way to harness the power of the BEAM. In this chapter, you'll learn how to join different Erlang nodes to form a cluster and the benefits of doing so. You'll deploy an application, lovingly written by our friend Ricardo García Vega, that uses Phoenix's Pub/Sub module to pass data between the multiple nodes in your Erlang cluster.

the (as yet) unwritten content

In this chapter, we'll discuss how to make your production environment more reliable by configuring your application to automatically scale based on the CPU usage of your remote servers. You'll optimize your production environment by adding an AWS Auto Scaling group and Load Balancer to your infrastructure. You'll learn how to ensure zero-downtime deployments by automatically rolling back your application upon failed deployments. You'll also understand how to improve the health of your production resources by cleaning up any unused artifacts any time you deploy.

the (as yet) unwritten content

Instrumentation is the last important step in the software delivery cycle, as it allows you to monitor your applications after they have left your machine and are deployed to production. You'll use Grafana, Promtail and Loki to log your application and Prometheus, and PromEx to collect the basic metrics produced by the main Elixir libraries in your Phoenix Application.

the (as yet) unwritten content

This chapter builds on the basic instrumentation that you implemented in Chapter 11 and discusses how you can implement you own custom metric for your Phoenix application. We'll also explore how to set up alerts for your application so that you're notified when unexpected issues arise in any of your environments.

the (as yet) unwritten content

This last chapter summarizes your learnings in this book. It will reinforce the different BEAMOps principles that we have discussed and how you can use them in your daily work to further your career as a multi-disciplinary engineer.

Any time we introduce a new technology in a chapter, we always take care to break it down for you. If getting too detailed is outside of the scope of this book, we'll always point you to other resources where you can find more information.

At the end of every chapter, you'll see a section titled "The Extra Mile." These sections include further exercises for you, if you're eager to apply what you've learned to novel scenarios. The essential content that you need to know is covered in the chapter, but these sections contain fun exercises that will give your brain a workout. The exercises are not guided, but we'll point you to the answers to the related questions.

Online Resources

The full source code for all the sample code used in this book, along with all images, can be found at the Pragmatic Programmers site for this book. There, you'll also find an errata page listing any updates and corrections made to the book, as well as a way for you to report any errata you may find.

That's enough chit-chat. Let's begin our journey!