# Завершение

Книга подошла к концу, а по большому счету мы только начали понимать концепцию Cloud Native и ее реализацию с помощью контейнеров и Kubernetes. Это и есть самое важное - понимая, что именно лежит в основе концепции, почему стали популярны микросервисы, что точно представляет собой контейнер, и какие принципы лежат в работе Kubernetes, можно уже с гораздо большей скоростью двигаться дальше, просто используя справочную информацию.

К примеру, мы вообще не затронули тему работу с данными - все наши простые примеры и обсуждения были основаны на микросервисах без состояния (stateless). Тем не менее любое приложение и система всегда обладает данными, которые нужно хранить. Дело в том, что управлять томами с данными, сетевыми дисками, масштабировать базы данных довольно тяжело, и в общем и целом это не является разработкой и программированием, а относится к инфраструктуре. Гораздо проще и надежнее в этом случае воспользоваться системой хранения данных, предлагаемых облаком. Это могут быть управляемые облаком классические базы данных (Amazon RDS, Google Cloud SQL), хранилища неструктурированных данных (Amazon S3), и много других решений, предлагаемых облаками. Это значительно упростит разработку и масштабирование вашей системы, это же рекомендуется 12 факторами (фактор 6).

Возвращаясь к 12 факторам облачного приложения, обсуждая их и работу с конфигурацией приложения, мы несколько раз упоминали, как Kubernetes упрощает конфигурацию с помощью секретов и карт конфигураций (config maps). Это важная часть, но работает она очень просто, и одного простого фрагмента YAML из любого подходящего примера на GitHub или сайта Kubernetes будет достаточно, чтобы начать ими пользоваться.

Мы также не обсудили работу с несколькими кластерами, контексты kubectl (context), и более эффективное использование ресурсов кластера с помощью разделенных пространств имен (namespaces). Можно снова сказать, что это важные части Kubernetes, однако они относятся больше к организации работы, а не к базовым концепциям, и их также будет проще найти в справочных страницах в Интернете.

Если вы еще помните начало книги, мы пообещали, что намеренно сделаем ее компактной и небольшой, и не станем перепечатывать документацию, примеры, существующие хорошие обучающие статьи, блоги, и все остальное. Если после прочтения этой книги вас по настоящему заинтересовал мир приложений Cloud Native, созданных для облака, как раз настало время идти дальше, воспользоваться местами очень хорошей онлайн-документацией, которая к тому же чуть не несколько раз в день обновляется и остается актуальной, найти подходящее вам по стилю обучающее видео, и продолжать свое путешествие вглубь страны Cloud Native.

Ресурсов и материалов огромное количество, не вся документация бывает высокого качества, но в целом два основных ресурса по Kubernetes и Docker предлагают отличную справочную информацию, весьма актуальную после того, как стали понятны основы концепции Cloud Native и контейнеров:

* kubernetes.io - центр информации, посвященный непосредственно Kubernetes. Не вся информация подается очевидно, некоторые статьи ссылаются друг на друга, но я надеюсь, после прочтения данной книги вам станет чуть легче ориентироваться в океане информации по Kubernetes.
* docker.io / docs.docker.com - основной сайт компании и технологии Docker и документация по технологии контейнеров и продуктам Docker. Надо сказать, что документация просто великолепная, навигация простая и очевидная, а раздел подробной справочной информации по командной строке docker и формату Dockerfile достойны того, чтобы быть постоянно открытыми в вашем браузере. Делает работу с контейнерами намного проще и эффективнее.
* www.cncf.io - главный сайт фонда Cloud Native Foundation. Это более обзорный сайт, но он дает неплохое понимание экосистемы, существующей вокруг концепции Cloud Native, например, посмотрите раздел проектов (projects). Можно многое узнать о способах работы с контейнерами, их безопасности, сборе журналов и метрик, и многом другом.
* Канал CNCF на YouTube содержит большую часть выступлений с конференций KubeCon, а также совместных конференций, посвященных различным смежных технологиям, таким как сервисные сетки service mesh. Некоторые видео и обучающие тренинги довольно удачно описывают многие концепции, если вам нравится данный формат.

В книге мы в основном, для простоты понимания общей картины, использовали самые базовые инструменты, такие как управление кластером с помощью команды kubectl, локальные кластеры minikube и docker, и только вскользь упоминали другие, расширенные инструменты. Чем больше вы станете программировать в стиле Cloud Native, чем очевиднее станут причины для использования сервисных сеток, шаблонов для конфигураций YAML, и много другого. Можно рекомендовать следующие инструменты, иногда значительно облегчающие работу:

* Skaffold, среда ускоренной постройки контейнеров с одновременным развертыванием в кластере Kubernetes. Создана с участием моей команды в Google, убирает постоянную необходимость заново помечать образы контейнеров новыми версиями и вызывать kubectl, вместо этого автоматически развертывая ваше приложение при любых изменениях в исходном коде. Если вы предпочитаете работать в редакторе IDE, посмотрите также на Google Cloud Code.
* Kubernetes Dashboard, стандартная визуальная панель мониторинга вашего кластера. Устанавливается в ваш кластер обычной командой kubectl apply, и значительно облегчает процесс знакомства с кластером и его ресурсами. Все детали на репозитории GitHub.
* Сервисные сетки (service mesh) Istio и Linkerd. Довольно высокий порог изначальной настройки и обучения, особенно Istio, но после установки и знакомства вы получаете огромные возможности по управлению трафиком и взаимодействию ваших микросервисов, и панели мониторинга за всеми вызовами, задержками, ошибками и всем остальным. Трудно оценить на маленькой “игрушечной” системе, лучше применять на реальной микросервисной архитектуре.

На этом наша краткая обзорная книга заканчивается. Надеюсь, что ваше дальнейшее знакомство с облачными приложениями будет продуктивным, программирование в стиле Cloud Native доставит массу удовольствия, а дизайн приложений в виде микросервисов станет намного проще. В результате должна получиться гибкая, мощная, устойчивая к отказам, всегда доступная система, легко масштабируемая под запросы миллионов пользователей по всему миру. Удачи!