**Для новичка, который знает Go:**

1. **Основы разработки веб-приложений на Go**:
   * Изучить основные фреймворки для веб-разработки на Go, такие как **Gin** или **Echo** (для создания API).
   * Изучить работу с **HTTP** и **JSON** (для работы с запросами и ответами).
   * Понять основы работы с **JWT** (для авторизации и аутентификации).
2. **Докер и контейнеризация**:
   * Изучить **Docker**, чтобы контейнеризировать сервисы и запускать их в изолированной среде.
   * Понять, как писать **Dockerfile** и как настроить **Docker Compose** для работы с несколькими сервисами.
3. **gRPC и REST**:
   * Понять, как работают **gRPC** и **REST** для взаимодействия между микросервисами.
   * Настроить простой **API Gateway** на Go, используя один из этих протоколов.
4. **HLS для стриминга видео**:
   * Изучить, как работать с **HLS** (для передачи видеофайлов через HTTP).
5. **Redis и очередь сообщений**:
   * Изучить основы **Redis** и как использовать его для кэширования и очередей (например, для асинхронных задач).

**Для новичка, который знает только основы Java:**

1. **Основы веб-разработки на Java (Spring Boot)**:
   * Изучить **Spring Boot** для создания RESTful API, так как это будет основной фреймворк для работы с микросервисами.
   * Понять работу с **REST** и **JSON** (для взаимодействия с API).
2. **JWT и безопасность**:
   * Понять, как работает **JWT** для авторизации и аутентификации, а также как защищать API.
3. **Работа с базой данных (PostgreSQL)**:
   * Изучить основы работы с **PostgreSQL** через **JPA** (Java Persistence API) для создания сущностей и выполнения запросов.
   * Разобраться с **миграциями** и настройкой базы данных.
4. **Docker**:
   * Аналогично с Go, изучить **Docker** для контейнеризации и использования с микросервисами.
5. **Интеграция с платежными системами (заглушка)**:
   * Понять основы интеграции с внешними сервисами, например, **Stripe** или **PayPal**, для симуляции работы с платежами.
6. **Работа с сервисами каталогов и подписок**:
   * Разработать сервис для управления подписками и каталогом фильмов, настроить поиск и фильтрацию данных.

**Общие шаги для обоих:**

1. **Swagger для документации API**:
   * Изучить, как использовать **Swagger** для документирования API.
   * Понять, как генерировать и поддерживать актуальные спецификации API.
2. **Тестирование**:
   * Изучить основы **юнит-тестирования** и **интеграционного тестирования** для каждого микросервиса.
   * Для фронтенда использовать **Jest** для написания тестов (если команда будет разрабатывать фронтенд самостоятельно).
3. **CI/CD**:
   * Изучить основы **CI/CD** (например, с использованием **GitLab CI** или **GitHub Actions**) для автоматической сборки и деплоя.