Приложение:

- https://github.com/soar/devops-test - как пример приложения для установки (также расписано что реально проверяют у людей)

Devops/SRE - это чаще всего не входная позиция, поэтому объем знаний для изучения велик. Тут нужна вовлеченность, не бойся и просто методично прорабатывай. Первое чему стоит научиться - не пробовать сожрать слона за раз. Проведи упражнение по работе с проектом и разбей на задачи. Веди дневник - что удалось, что не удалось, почему, что узнал нового. Лучше веди его в Git в виде markdown записей - и перед новой задачей занимайся саморефлексией

Начни с теор азов - почитай про SDLC и как вообще строится процесс поставки качества (value)&
Computer Science Center - Разработка ПО (весна 2021) на YouTube

Примерное время исполнения: 2-3 месяца

Работа с проектом:
- В любом удобном месте (можно попробовать Ya.Tracker) разбить работы на задачи (максимальное время на задачу - 1нед., оптимальная - 2-3 дня)
- Во время проекта вести краткое описание выполненных работ с трекинго времени
- Почитать про спринты и канбан (Agile подход)

Что надо сделать:

Режим админа - base lvl:

1. Установить указанный сервис в 2 инстанциях + 1 инстация БД (развернуть подходящие виртуальные машины)
2. Поставить балансировщик на свое усмотрение (NGINX, Envoy, HaPROXY) и настроить чтобы он открывался через него
3. Настроить ротацию логов (logrotate) и бекапирования БД (время запуска с 00:00 - 04:00:
    1. Убедиться, что в остальное время бекапирование не срабатывает, чтобы не было нагрузки для БД и влияния на пользователей
    2. Убедиться, что нет других процессов бекапирования, чтобы не создавать лишние блокеры на базе
    3. (*) Написать скрипт верификации бекапа (развертываем новую тестовую базу + тестовую инстанцию и проводим Sanity тестирование) - артефакт либо автотесты, либо коллекцию Postman-а

Режим devops - good lvl:

1. Развертывание должно происходить автоматически (Как пример с помощью Ansible) и с минимальным простоем
2. Для развертывания использовать не голую ВМ - поставить и настроить Docker + создать Dockerfile и настроить доставку образа
3. CI удобным для себя способом (поставить GitLab/Jenkins, использовать GitHub)
    1. Сборка и тестирование
    2. Упаковка и доставка до хранилища артефакта (если решил без Docker - то к себе на шару, если Docker - то в Docker repository)

Режим devops - nice lvl:

1. Написать docker-compose для локального развертывания продукта локально у себя и разработки
2. Развертывание должно происходить с временем близким к Zero-Time (BG , Rolling Update + прогон Postman/Ciprus тестов на развернутой машине)
3. В случае сломанного деплоя - откат на предыдущую версию по апруву
4. К CI добавляем CD:
    1. Развертывание Docker-compose инсталляции + прогон Sanity/Smoke
    2. По апруву залитие на PROD стенд

Мониторинг:
1. Поставить Grafana + Prometheus
2. Поставить node_exporter на ноды
3. Взять готовый борд с Grafana Market под linux (посмотреть и разобраться, что рисуется)
4. Настроить базовый хелчек по состоянию работает/не работает сервис (blackbox_exporter)
5. Поставить alertmanager + настроить алерты (читаем про RED/USE)
    1. Что весь сервис лег (сервис на обеих нодах не работает)
    2. Базовые алерты по некорректным состояниям (рекомендую обратить на букву S в USE - Saturation)
6. (*) Добавить отправку метрики из CI (версия, состояния процесса поставки, время деплоя) - почитать про DORA метрики

Сбор логов:
1. Опционально, обычно стоит ElasticSearch + сборщики разного типа (Logstash, Filebeat, Vector, Telegraf)
2. Rsyslog

Суть в том, чтобы централизовано хранить структурированные (или не полность структурированые) данные и иметь полнотекстовый поиск по ним

Автоматизация:
1. Написать инструмент (скрипт) для CI который отправляет уведомление об статусе деплоя в Telegram + ждет апрува от списка апруверов и информирует если базовый ХЧ не прошел с просьбой либо апрувить дальнейший накат, либо роллбек - рекомендую Python, либо Go (https://quii.gitbook.io/learn-go-with-tests/)

Опционально:
- Зарегаться на GitHub - проект под автоматизацию вести там.

Неопционально:
- git (у мелкомягких есть неплохой - https://docs.microsoft.com/en-us/learn/modules/intro-to-git/)

Что изучаем:
- SDLC (Software Development LifeCycle)
- Git
- Системы хранения кода и построения процессов SDLC (Gitlab, Github, BB, Jenkins, TeamCity, etc.)
- Linux (/dev, /proc, namespaces, strace, dtrace, iotop, htop, lsof и прочее)
- Monitoring Stack (Prometheus + Alertmanager + Exporters обязательно, VictoriaMetrics, InfluxDB + Telegraf - для расширения кругозора)
- Containers (Docker - as must, Podman - опционально) - https://www.redhat.com/sysadmin/cgroups-part-one, https://aws.plainenglish.io/containers-from-zero-to-hero-2-namespace-and-cgroups-eaf587396749, Docker доки и Podman доки
- K8s (возможно поставить minikube) и попробовать туда задеплоиться - сейчас основная система оркестрации вычислительными мощностями через управление контейнерами - дока по k8s, на medium ищем статьи по более глубокуму описанию
- Централизованная сборка логов (ELK, Filebeat, Vector, Telegraf)
- kvm/proxmox - https://pve.proxmox.com/pve-docs/ , почитать за VMWare/Openstack
- Python или Go для автоматизации под Linux (можно попробовать написать базовый REST, бота), PowerShell под Windows
- Архитектура приложений (Monolith, microservices, REST, GRPC, Service Discovery)
- Основные протоколы взаимодействия (GRPC, HTTP 1.1/2, QUIC, TCP, UDP, BGP)
- Безопасность (AppArmor, Seccomp, OWASP 10) - неплохо ознакомиться и понимать, что придется защищать
- Балансировка - в Medium ищем

Список источников:
- Medium - там сейчас можно найти все
- https://hpbn.co - про HTTP

- https://microservices.io/index.html - за микросервисы
- https://martinfowler.com/delivery.html- за софт и SDLC
- https://agilemanifesto.org/iso/ru/manifesto.html - Agile манифест, как небольшая накачка
- https://brendangregg.com - за системный перформанс и мониторинг
- Linux - Кетов в Youtube + книжка, затем маны системы
- istio - самый популярный service mesh
- https://github.com/binhnguyennus/awesome-scalability - как делать хорошие системы с возможностью к масштабированию
- https://github.com/DevOpsStuff/Introduction-to-DevOps - Roadmap для Devops
- https://medium.com/javarevisited/the-2018-devops-roadmap-31588d8670cb - в догонку
- https://www.youtube.com/watch?v=Rryi_1JZnuc - про то, что система не черный ящик
- https://github.com/mxssl/sre-interview-prep-guide

На собесах:
- Не надо пробовать казаться круче чем есть - лучше честно рассказать что делал, что читал и слышал но не трогал, куда хочешь двигаться
- Перед собесом забей в гугле top devops questions
- Hackerrank - решай задачки, время Yaml погроммистов уходит - даже для базовой автоматизации нужны знания разработки
- Проработай вопросы с наблюдаемостью - подумай как бы ты мониторил систему, на что бы обращал внимание