Задание 1.

1.Компонентную схему контейнеров, с обоснованием декомпозиции с точки зрения отказоустойчивости

**Python приложение**: Развертывание в отдельных контейнерах для каждой инстанции приложения. Рассмотреть возможность обновления до Python 3., учитывая что Python 2.7 не поддерживается.

**Redis**: Контейнеризация с использованием официального образа Redis. Для отказоустойчивости предусмотреть кластеризацию Redis с использованием Sentinel.

**MySQL**: Развертывание двух независимых контейнеров для master-master репликации. Использовать официальные образы MySQL. Рассмотреть внедрение Orchestrator для управления репликацией и отказоустойчивости.

**MySQL Proxy**: Контейнер для проксирования запросов к MySQL. Это позволит обеспечить прозрачное распределение нагрузки и упростит масштабирование.

**Apache и Nginx**: Развертывание в отдельных контейнерах. Apache может использоваться для обработки динамического контента, в то время как Nginx может выступать в роли reverse proxy и обрабатывать статический контент и кеширование.

Всё в docker image и для всего каждого свой volume.

***Обоснование***

**Отказоустойчивость**: Разделение на микросервисы в контейнерах упрощает процесс достижения высокой доступности и отказоустойчивости, так как каждый компонент может быть независимо масштабирован и восстановлен без воздействия на остальную часть системы.

**Масштабируемость**: Декомпозиция позволяет масштабировать наиболее нагруженные компоненты системы, не затрагивая остальные.

**Обновления**: Контейнеризация упрощает процесс развертывания обновлений и новых версий компонентов, минимизируя простои и риски.

**Управление конфигурацией**: Хранение конфигураций в volume'ах или внешних хранилищах упрощает управление и обновление конфигураций без необходимости

2. Описание образов необходимых контейнеров

#### ***Python приложение***

Базовый образ: python:2.7-alpine

в Dockerfile

- Установка зависимостей приложения через pip.

- Копирование исходного кода приложения в образ.

***Redis***

Базовый образ: redis:alpine

Настройка Redis для работы в кластерном режиме, если это необходимо.

***MySQL***

Базовый образ: mysql:latest

Настройка my.cnf для master-master репликации.

Volume для баз данных

***MySQL Proxy***

Базовый образ: alpine

Основные шаги:

- Установка MySQL Proxy.

- Настройка проксирования запросов к MySQL.

***Apache***

Базовый образ: httpd:alpine

если требуется volume.

***Nginx***

Базовый образ: nginx:alpine

если требуется volume.

3.Подробное описание процесса развертывания на новом хосте

1. Всё разворачивать будем с использованием ansible, напишем нужные роли

2. Установим Docker и Docker compose

3. настроим права для доступа к файлам

и запуская роль в ansible будем запускать или docker контейнеры

или написать docker compose файл для запуска сразу всего одним файлом