Задания не связаны между собой.
Помеченные звездочкой задания имеют повышенную сложность.
Не обязательно выполнять все задания, выберите те, которые сможете реализовать.
Можно использовать один проект на `github` для предоставления результатов с каталогами под каждое из заданий.

Задание 1

Напишите `ansible`-плейбук, для конфигурации веб-сервера `nginx`:

- для выполнения задачи используйте подходящие модули `ansible`
- выбор операционной системы на ваше усмотрение
- плейбук должен устанавливать пакет `nginx`, настраивать его конфигурационный файл, производить запуск веб-сервера, проверять его доступность по `80` или `443` порту
- используйте параметры (`variables`) и шаблоны (`templates`) везде, где это возможно
- приложите скриншот или лог выполнения плейбука
- предоставьте результат в виде кода плейбука или всего проекта `ansible`, загруженного в проект на `github`

Задание 2

Напишите `dockerfile` и `docker compose` для него, обеспечивающий запуск веб-севера `nginx`:

- `dockerfile` должен включать в себя добавление конфигурационного файла `nginx`
- конфигурационный файл `nginx` должен:
  - обеспечивать работу веб-сервера на `443` порту
  - использовать самоподписанный сертификат для доступа по `https`
  - иметь правило по умолчанию для редиректа с `80` на `443` порт
  - веб-сервер должен отдавать статическую `html` страницу с любым содержимым на ваш выбор
- в состав итогового `docker` образа (`image`) должны входить:
  - конфигурационный файл `nginx`
  - файл самоподписанного сертификата и его ключ
- `docker compose` файл должен обеспечивать:

    - работу контейнера на основе собранного образа из `dockerfile` подговтоленного в прошлом шаге
    - автозапуск контейнера при рестарте операционной системы
    - статический `html` файл для `nginx` должен находиться на диске хоста и добавлен в контейнер как `volume` (из локальной директории или заранее созданного `volume`)
    - открытый порт для доступа к `nginx` с локального хоста
- предоставляемый результат должен включать в себя файлы, загруженные в проект на `github`:
    - `dockerfile`
    - `docker-compose.yaml`
    - самоподписанный сертификат и его ключ
    - статический `html` файл для `nginx`
    - скриншот или текстовой файл с результатами выполнения команды `curl` обращающегося к вашему `nginx`, запущенному в контейнере (например: `curl -vk https://localhost:443`)

Задание 3

Напишите скрипт используя `bash` , который обеспечит:

- проверку состояния диска на занятый объем в процентах (можно любого диска или раздела в системе)
- если свободного объема менее `85%`, отправит уведомление (алерт) на почту
- параметризацию настроек для адреса `smtp` сервера, логина и пароля к нему

Используйте любую удобную реализацию отправки почты через `smtp` (клиент на ваш выбор).
Для отравки почты используйте любой публичный `smtp` сервер, например вашей личной почты на `yandex`, `mail`, `gmail`.
Предоставьте результат в виде сценария, загруженного в проект на `github`.
Не добавляйте в результат ваши личные данные для авторизации на `smtp` сервере!

Задание 4 ⭐

Напишите пайплайн для сборки и доставки приложения (`CI/CD`).
Вы можете использовать любую из возможных и доступных вам `CI` систем.
Публичные сервисы типа `GitHub (GitHub Actions)` или `GitLab` представляют свои инструменты бесплатно, их возможностей должно быть достаточно для выполнения данной задачи.

Этапы пайплайна (сборка и доставка) могуть быть реализованы в любом удобном для вас формате:

- сценарий `bash`
- `ansible` плейбук
- любые профильные инструменты предназначенные для подобных задач

Этап сборки (`CI`) должен включать в себя создание `docker` контейнера (можно использовать результат из **задания 2**).
Этап доставки (`CD`) должен обеспечивать запуск приложения из созданного `docker` контейнера.

Предоставьте результат в виде сценариев (пайплайнов) и результата их выполнения (в виде скриншотов или лога выполненных этапов пайплайна), загруженных в проект на `github`.

Задание 5 ⭐⭐

Воспользуйтесь бесплатным периодом `Yandex Cloud` и создайте облачную инфраструктуру используя `terraform` отвечающую этим требованиям:

- используйте провайдер `terraform` для Yandex Cloud
- опишите создание виртуальной машины и необходимых для ее работы зависимостей (сети, диски и прочее)
- не обязательно: определите ресрусы облака как отдельные модули `terraform` и используйте их в своем сценарии

Для работы с `terraform` вам может понадобиться `proxy` или `vpn` для доступа к документации и провайдерам (ресурсы заблокированы для доступа из региона РФ).
В документации `Yandex Cloud` вы сможете найти инструкцию по настройке использования их зеркала с ресурсами `terraform`.

Предоставьте результат в виде `terraform` файлов и результата их выполнения (в виде скриншотов из панели управления облаком), загруженных в проект на `github`.
Не добавляйте в результат ваши личные данные для авторизации с облаком!