Momo Store

momo.kuropatko.ru

Использованный stack

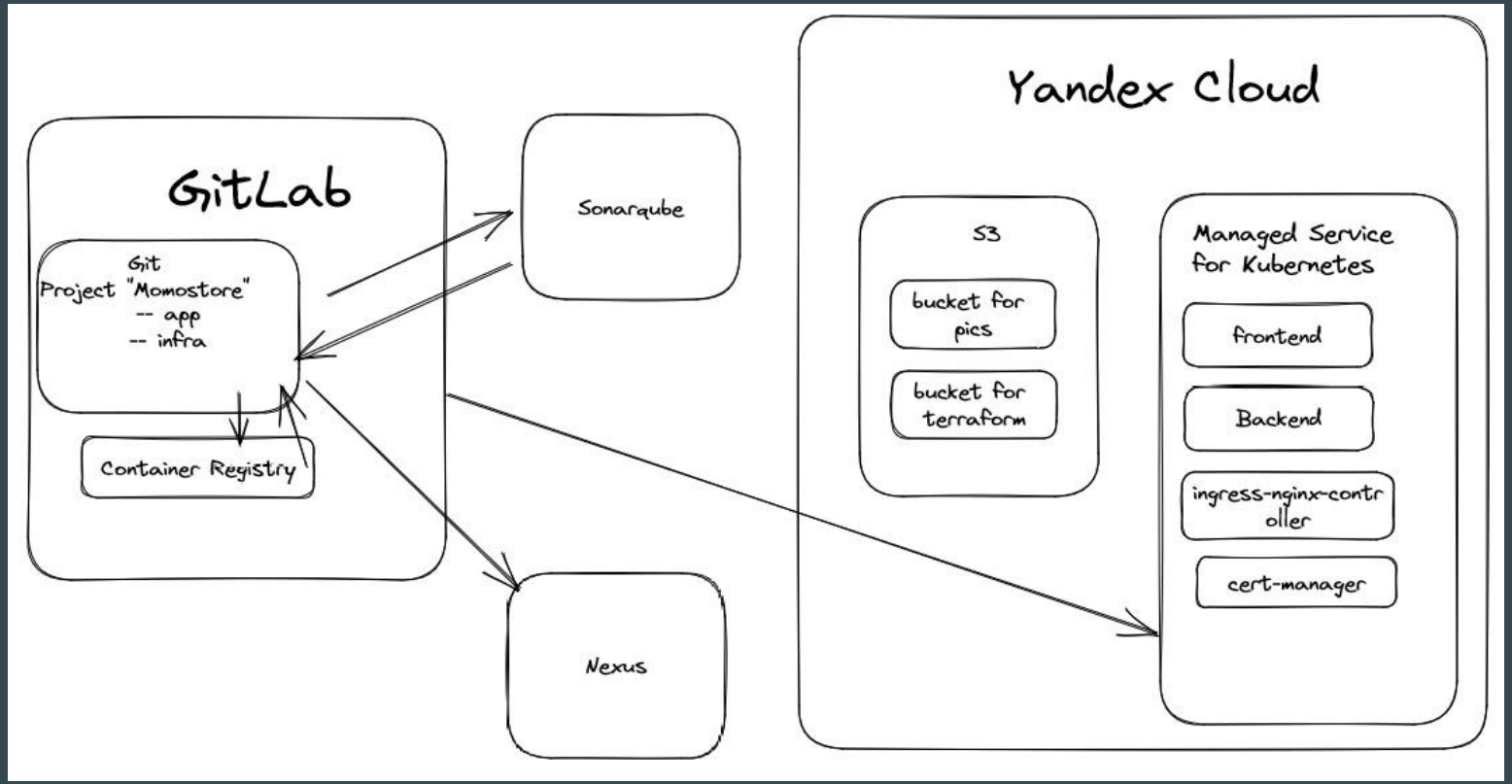
- Gitlab - хранение кода инфраструктуры и приложения, а также реализация CI/CD-процессов.
- SAST, Sonarqube, "go test" - тестирование исходного кода и собранного приложения.
- Docker - сборка образов приложения для запуска в docker-контейнерах.
- Kubernetes (YC Managed Service for Kubernetes) - среда (кластер) для работы нашего контейнеризированного приложения.
- Helm - Шаблонизация kubernetes-манифестов нашего приложения. Хранение версий. Деплой приложения в кластере.
- Nexus - Репозиторий для хранения версий helm-пакетов приложения.
- S3 buckets - Хранение статики сайта и состояния Terraform.
- Terraform - Описание используемой для приложения инфраструктуры (IaC).

Ресурсы Яндекс Практикум

Использованные уже имевшиеся ресурсы от Яндекс Практикум:

- Gitlab
- Nexus
- Sonarqube
- VM (ВМ в роли агента, для запуска конфигураций terraform, а также отладки сценариев docker/docker-compose, terraform, yc, kubectl, bash/shell и т.д.

Схема инфраструктуры приложения



Хранение проекта в git-репозитории. Структура

В **Gitlab-репозитории** создан проект **momostore**. Код приложения, как и код инфраструктуры хранится в одном проекте.

Container Registry Gitlab использован как для хранения образов каждой версии приложения (разделен на frontend и backend), так и вспомогательных образов для работы в пайплайнах (образы с golang, node, helm-kubectl, alpine итд)

- директории frontend и backend - исходный код приложения+dockerfile
- директория momo-store-chart - helm-chart приложения
- директория terraform/S3buckets - конфигурации развёртывания бакетов в объектном хранилище YC

CI/CD pipelines

От сборки и тестирования до развёртывания приложения в кластере kubernetes приложение проходит два пайплайна и пять работ (job-ов).

Пайплайн 1 (Downstream):

1. Тестирование (SAST, Sonarqube, go-tests)
2. Сборка (в docker image, хранение gitlab docker registry)
3. Обновление версии helm-chart

Пайплайн 2 (trigger):

1. Развёртывание приложения в кластере kubernetis с помощью helm (deploy)
2. Отправка новой версии helm-chart приложения в репозиторий Nexus (https://nexus.praktikum-services.tech/service/rest/repository/browse/008-12-Helm/)
   Реализована схема module-pipelines для разделения работ по частям приложения. Если сделан коммит в в код в файлах в директории./backend, то стандартный цикл работ запускаются только по backend и аналогично то же самое с frontend.

Установка актуальной версии helm-chart приложения (версия hel-чарта и версия приложения) служит триггером для запуска пайплайна развёртывания приложения в кластере.

Тестирование SAST, Sonarqube

- В пайплайне предусмотрено статическое тестирование безопасности приложений. **SAST** запускается на этапе **test**.

- Автоматический анализ кода отдан внешнему **Sonarqube**. На этапе **test** Job выполняет команду sonar-scanner , которая обращается к файлу конфигурации и следует его инструкциям.

Релизный цикл/версионирование

Подразумевается классическая схема:

- S - минимальные изменения, как правило, исправление багов (изменение третьей цифры номера версии)
- M - более серьезные изменения, в частности, правка БД (изменение второй цифры номера версии)
- L - действительно масштабные изменения (изменение первой цифры номера версии)

В нашем случае используется схема "S", то есть 1.0.<версия>

Третье значение привязано к номеру пайплайна (переменная CI_PIPELINE_ID).

Версия назначается тегу docker-образа и версии helm-chart (общая версия+версия части приложения - backend или frontend)

Версии "M" и "L" меняются в описании пайплайна руками, после например оценки важности и масштабности вносимых изменений.

Helm-chart и версии внутри версионируются с помощью скрипта **helmchart_ver.sh**.

Инфраструктура

Для реализации кластера kubernetes выбран сервис Yandex Cloud "Managed Service for Kubernetes"

Для реализации S3-хранилища и необходимых бакетов выбран сервис Yandex Cloud "Object Storage"

Конфигурация описана в terraform и хранится в git проекта.

Инфраструктура: сценарий TF для kubernetes

- указание бакета для хранения состояния
- необходимые аккаунты для работы отдельных сервисов (managed kubernetes, ingress)
- назначение аккаунтам ролей
- генерация ключа
- сетевая структура (рабочая сеть, подсети)
- сетевые доступы и правила
- группы безопасности
- описание характеристик создаваемых worker nodes

В качестве ingress-контроллера развёрнут ingress-nginx-controller.

Для использования SSL/TLS сертификата развёрнут cert-manager.

Инфраструктура: сценарий TF для S3

- указание бакета для хранения состояния
- создание сервисного аккаунта
- назначение ролей сервисному аккаунту
- создание статического ключа
- создание необходимых бакетов и политик доступа

Скрипт yc_kubectl_configng.sh служит для настройки с помощью CLI YC kubeconfig и доступа к кластеру kubernetes.

Развёртывание инфраструктуры выполняется с VM-агента руками, но вполне может быть встроена в пайплайн CI/CD.

Deploy (развёртывание) приложения и хранение helm-chart пакетов

Развёртывание приложения выполняется в пайплайне после изменения версии helm-chart приложения актуальную.

Для деплоя используется в раннере образ helm-kubectl:

1. Конфигурируется с помощью CLI YC и переменных (чувствительные данные) Gitlab CI kubeconfig
2. С помощью helm обновляется приложение в кластере
3. Далее в helm-репозиторий Nexus выгружается актуальная версия helm-chart приложения