

Санкт-Петербургский государственный университет

Кафедра системного программирования

Группа 22.Б15-мм

Расширение пользовательских сценариев и функциональных возможностей IaC-платформы infractl

ЯКШИГУЛОВ Вадим Наилевич

Отчёт по производственной практике
в форме «Решение»

Научный руководитель:
к. ф.-м. н., доц. Е. К. Куликов

Консультант:
руководитель службы, ООО «Яндекс.Технологии» А. В. Лоптев

Санкт-Петербург
2025

Оглавление

Введение	3
1. Постановка задачи	5
2. Описание терминов и базовых команд infractl	6
2.1. Термины	6
2.2. Общая архитектура	7
3. Определение запросов пользователей	10
4. Формирование требований	12
4.1. Создание групп безопасности	12
4.2. Частичная пересборка артефактов	12
4.3. Команда diff для фермы балансеров	13
5. Описание реализации	14
5.1. Создание групп безопасности	14
5.2. Частичная пересборка артефактов	16
5.3. Команда diff для фермы балансеров	16
6. Адаптация	18
Заключение	19
Список литературы	21

Введение

Современные облачные решения стремительно развиваются и уже сейчас включают в себя такие инфраструктурные компоненты (конечные системы), как балансировщики нагрузки, системы мониторинга, инструменты трассировки и средства развёртывания приложений. Однако с ростом сложности всё больше пользователей сталкиваются с проблемами настройки, включая устаревание функциональности, несовместимость с изменёнными API и проблемы воспроизводимости операций при императивном подходе при создании компонентов.

Для упрощения управления инфраструктурой разработаны решения типа IaC (Infrastructure as Code), такие как Terraform[3], и сервисы управления разнородной инфраструктурой, такие как Crossplane[1], которые позволяют декларативно описывать инфраструктуру и упрощают управление ею. На основе аналогичного подхода создана платформа infractl¹, предназначенная для использования внутри Яндекса. Данная платформа ориентирована на эксплуатацию крупных систем, работающих в условиях повышенных требований к надёжности, масштабируемости и устойчивости к высоким нагрузкам.

Платформа предоставляет пользователям возможность управления инфраструктурными компонентами с помощью CLI и поддерживает: импорт конфигурации из конечных систем, создание декларативных описаний с нуля, сборку артефактов и многие другие функции. Однако, в связи с большим количеством различных настроек, исторически сложившимися особенностями и широким разнообразием уже существующих компонентов, текущая функциональность не всегда удовлетворяет потребности пользователей, что снижает эффективность управления инфраструктурой, уменьшает популярность платформы и может даже приводить к возникновению инцидентов.

Таким образом, цель работы заключается в выявлении и устранении актуальных проблем, а также в улучшении взаимодействия пользователей с платформой, что должно сделать процесс управления инфра-

¹<https://habr.com/ru/companies/yandex/articles/854506/> (дата обращения: 21.12.2024)

структурой более простым и понятным.

1 Постановка задачи

Целью работы является повышение эффективности и удобства IaC-платформы infractl через реализацию востребованных функций.

Для достижения поставленной цели были поставлены следующие задачи:

1. Определить актуальные запросы пользователей платформы.
2. Сформировать требования для каждой из выявленных потребностей.
3. Реализовать новые функциональные компоненты в соответствии с требованиями.
4. Проанализировать влияние внесенных изменений на пользователей.

2 Описание терминов и базовых команд `infractl`

Для полноценного понимания основных инфраструктурных компонентов, с которыми работает платформа `infractl`, а так же доступных возможностей предоставим их краткий перечень.

2.1 Термины

2.1.1 Общие

- Kubernetes (k8s)[2] — это платформа с открытым исходным кодом для автоматизации развёртывания, масштабирования и управления контейнеризированными приложениями.
- Ресурс — это объект или операция, доступ к которым осуществляется через Kubernetes API.
- Контроллер — это компонент, который следит за состоянием ресурса и выполняет различные действия для поддержания желаемого состояния.
- Инфраструктурный компонент (конечная система) — это внешний сервис, API которого используется контроллером для обновления информации о ассоциированном ресурсе.

2.1.2 Система сборки

- Артефакт сборки (исполняемый ресурс) — это файл или директория вместе с набором некоторых метаданных, создаваемые системой постановки распределенных задач.
- Подстановки — файл, которых хранит информацию о версии артефакта сборки. Обычно генерируется с нуля при запуске команды `build`.

2.1.3 Система управления подсетями

- Группы безопасности[5] — это ресурс, который создается на уровне облачной сети и используется для разграничения сетевого доступа объекта, к которому она применяется.

2.1.4 Система балансировки нагрузки [4]

- Балансировщик нагрузки (балансер) — это специализированное устройство или программное решение, предназначенное для равномерного распределения входящего трафика среди нескольких серверов или сервисов.
- L7-балансер — балансировщик нагрузки, работающий на прикладном уровне модели OSI. Он может принимать решения о балансировке нагрузки на основе контента, включая URL-адреса, заголовки HTTP или файлы cookie.
- L3-балансер — балансировщик нагрузки, работающий на сетевом уровне модели OSI. Он распределяет трафик между серверами на основе IP-адресов источника и назначения.
- DNS-записи — это пара из домена и ассоциированного с ним IP-адреса.
- Namespace — это абстракция конечной системы, которая может включать в себя информацию о L3, L7-балансерах, DNS-записях и других объектах, необходимых для балансировки.
- Ферма балансеров — это абстракция infractl над неймспейсом, упрощающая описание конфигурации и управление балансировщиками и DNS-записями.

2.2 Общая архитектура

На рисунке 1 представлена общая архитектура платформы. Зеленый и синий блоки — компоненты, написанные и используемые внутри

платформы, а фиолетовый блок — конечные системы, которые управляются infractl.

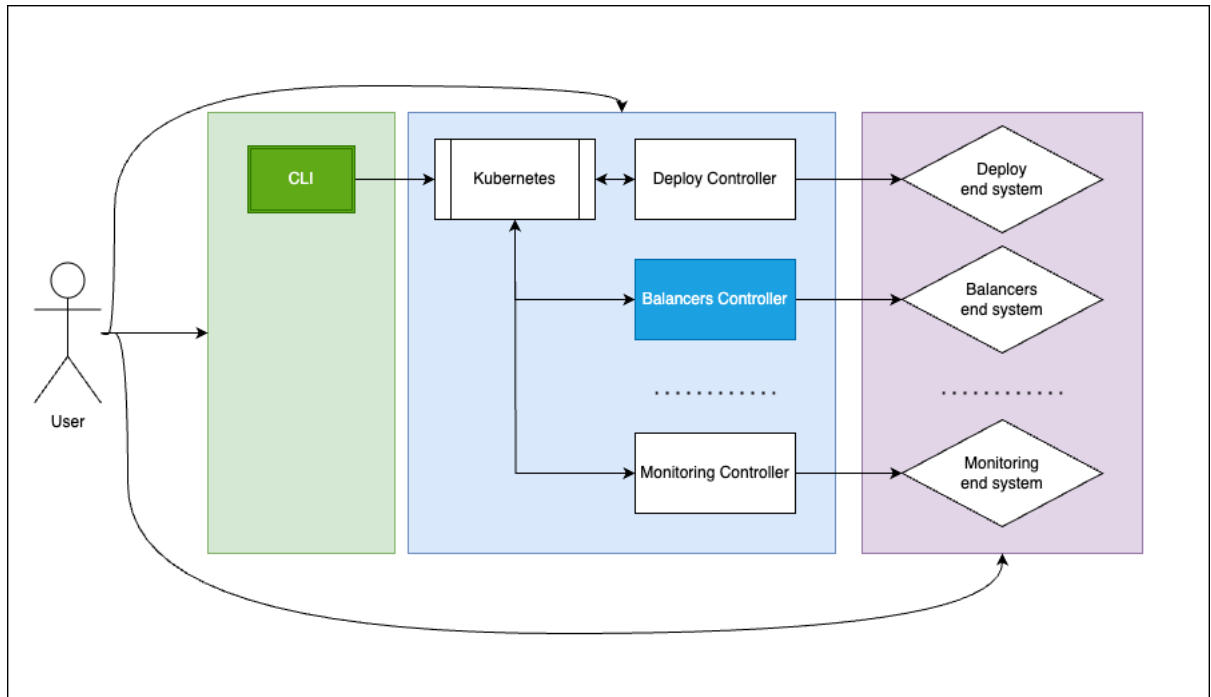


Рис. 1: Общая архитектура платформы

Платформа устроена так, чтобы отказ одного из компонентов не приводил к невозможности управления инфраструктурой. Так в случае недоступности или ошибки в CLI всегда можно поменять ресурсы с использованием стандартных средств kubernetes, например, kubectl. Если распределенный полокационный кластер kubernetes вдруг откажет, пользователь сможет поменять настройки своего проекта напрямую в конечных системах. При этом ручные изменения не будут перезаписаны в конечных системах после восстановления кластера из-за несоответствия спецификаций и фактического описания в конечной системе.

В рамках данной работы будут внесены изменения в CLI и контроллер балансеров, они отмечены темно-зеленым и темно-синим цветом на рисунке 1, рассмотрим их основные особенности.

CLI состоит из нескольких десятков команд, ниже приведены наиболее популярные из них.

- deploy — команда, которая обрабатывает переданную специфика-

цию инфраструктурных компонентов и отправляет информацию об изменениях в конечную систему через контроллер.

- `diff` — команда, которая показывает разницу между тем, что находится в конечной системе и тем, что пользователь отправит, используя команду `deploy` на переданной спецификации.
- `build` — команда для сборки необходимых артефактов (исполняемых ресурсов и Docker образов) в соответствии со спецификацией и сохранение информации о них в файл подстановок.
- `import` — команда для переноса объекта или группы объектов из конечной системы под управление `infractl`.

Контроллер балансеров, в свою очередь, предоставляет стандартизированный API для управления различными компонентами системы балансировки нагрузки: L7, L3-балансерами, фермами балансеров, DNS-записями и другими компонентами этой конечной системы.

При этом объекты, которые описаны в `infractl` могут как незначительно, например, как L3, L7-балансеры, отличаться от представления в конечной системе, так и очень сильно для упрощения описания их пользователем, как фермы. Такие различия часто приводят к необходимости нетривиальной трансляции из упрощенного варианта в более сложный и полный.

3 Определение запросов пользователей

Как у любого хорошего сервиса, у infractl есть чат поддержки для решения возникающих вопросов и для помощи пользователям, начинающим работу с платформой. Иногда такие вопросы завершаются созданием задач в трекере задач на исправление багов или введения новой функциональности. За такие задачи пользователи могут голосовать в интерфейсе трекера, так команда понимает, какие задачи вызывают ажиотаж.

Другим способом получить информацию от пользователя является прямой запрос к менеджерам или разработчикам infractl. Часто это большие команды, которые переносят части своей инфраструктуры, например, балансеры под управление infractl. И для них важно, чтобы этот переход произошел как можно более бесшовно. Обычно для таких команд организуется отдельный канал связи для доработки платформы под задачи конкретной команды.

Так был проанализирован чат поддержки за последние три месяца, а также задачи в трекере за этот период. Среди наиболее популярных по количеству лайков оказались следующие запросы:

1. Добавить флаги в команду `deploy` для создания групп безопасности.
2. Запретить в системе версионирования платформы переводить версии из статуса BETA в статус STABLE без создания промежуточной версии.
3. Валидировать наличие квоты доступных ресурсов (CPU, RAM и других) при создании фермы балансировщиков.

Чат поддержки содержал в основном вопросы о нюансах использования существующей функциональности. Однако было четыре сообщения от разных пользователей с запросом о пересборке только указанных артефактов — сейчас можно было либо пересобрать все артефакты, либо указать списком конкретные подстановки, а остальное пересобрать.

Отсутствие этой функциональности замедляло работу SRE во время инцидентов.

Кроме того, к техническому менеджеру проекта обратилась команда из одного подразделения компании для переноса 55 балансеров под управление infractl в качестве пилотного проекта. Для этого требовалось выполнить следующие задачи:

1. Реализовать diff для фермы балансеров, включая все дочерние объекты.
2. Оценить с помощью этого инструмента отличие импортированных балансеров от тех, что находятся в конечной системе.
3. Доработать diff и import для уменьшения отличий между объектами.

В результате обсуждения с командой было решено сделать задачи по созданию групп безопасности, частичной пересборки артефактов и команды diff для фермы балансеров, поскольку эти задачи были признаны наиболее приоритетными.

4 Формирование требований

На основе анализа пользовательских запросов были сформированы требования к новым компонентам платформы infractl. Все выявленные задачи можно условно разделить на три направления: создание групп безопасности, реализация частичной пересборки артефактов и разработка команды diff для работы с фермой балансировщиков.

4.1 Создание групп безопасности

Так как большинство команд в CLI используется, как в CI/CD и API внешних систем, так и напрямую пользователем из консоли, было решено добавить как интерактивный режим, так и неинтерактивный для данной функциональности.

Требования для реализации следующие:

- Интерактивный режим: платформа должна запрашивать данные у пользователя пошагово — на каждом этапе создания группы.
- Неинтерактивный режим: для работы через CLI необходимо предоставить следующие флаги:
 1. Флаг для выбора существующей группы.
 2. Флаг для создания новой группы.
 3. Флаг для опционального указания родителя для новой группы.

Реализация этих функций должна быть интегрирована в команды, где уже существует возможность выбора групп, включая команду deploy.

4.2 Частичная пересборка артефактов

Основным требованием к этой функциональности является сохранение обратной совместимости для текущих пользователей платформы. Поэтому было решено:

- Не менять поведение существующего флага `--incremental`, так как он уже активно используется в скриптах пользователей.
- Добавить новый флаг `-A`, который будет менять поведение команды на «пересобрать только переданные ресурсы».

Это упростит пересборку отдельных артефактов, устраняя необходимость доработки пользовательских скриптов.

4.3 Команда `diff` для фермы балансеров

Известно, что ферма балансировщиков включает в себя L3 и L7-балансировщики, неймспейсы и DNS-записи. Поэтому для реализации команды `diff` потребуется сравнение состояния для каждого из этих ресурсов, а также их интеграция в команде для фермы. Однако, поскольку заказчики не используют DNS-записи, было решено исключить их поддержку из пилотной версии.

При этом команда должна быть построена на базе существующих контроллеров платформы. Это обеспечит унификацию поведения и снизит риск расхождений между результатами выполнения `diff` и фактическими изменениями, вносимыми в конечную систему.

В случае, если один из контроллеров дочерних объектов фермы отказал, важно, чтобы остальные могли продолжать работу.

5 Описание реализации

5.1 Создание групп безопасности

Конечные системы, как правило, предоставляют несколько видов API, и при наличии выбора мы отдаем предпочтение протоколу Protobuf, затем REST, и лишь в последнюю очередь рассматриваем другие варианты API. В системе управления подсетями реализованы два типа API: устаревший REST и новый, построенный на основе GraphQL.

Использование устаревшего API было признано нецелесообразным, так как клиент для системы управления подсетями разрабатывался впервые, а ранее эта система не применялась в проекте. Поэтому был реализован клиент для GraphQL API с поддержкой функциональности создания групп безопасности и механизма автоматических повторов запросов (ретраев) для повышения отказоустойчивости конечной системы.

Получения списка всех групп безопасности обращения продолжают идти в сервис-прокси, так как он предоставляет более удобный API для этого. Так все группы безопасности, введенные пользователем, валидировались в CLI на наличие в сервисе-прокси. Однако на синхронизацию между сервисом управления подсетями и прокси требуется несколько минут, что неприемлемо для быстрого создания новой группы. Была реализована логика двойной валидации: если группы нет в прокси, то проверяется наличие в настоящем сервисе.

Интерактивный выбор создания новой группы был встроен в функциональность выбора и фильтрации группы, что сделало интерфейс более понятным. Рассмотрим вывод CLI при создании новой группы безопасности.

```
→ Enter security group: [Use arrows to move, type to filter]
> <create security group>
  sg_infractl_back
  sg_infractl_test
```

```
sg_infractl_prod
...
```

После выбора опции `<create security group>` предлагается ввести название новой группы. Значение по-умолчанию предлагается на основе названия сервиса.

```
→ Enter security group: (default: "sg_infractl_new_app")
```

Выбор родительской группы не позволяет создать группу для упрощения взаимодействия с командой.

```
? Do you want to attach newly created security group to some
existing parent group? Yes
→ Enter security group: [Use arrows to move, type to filter]
> sg_infractl_back
  sg_infractl_test
  sg_infractl_prod
  ...
```

Для неинтерактивного режима были добавлены флаги:

- `--sec-group=STRING` для выбора группы.
- `--new-sec-group=STRING` для создания группы с переданным именем.
- `--parent-sec-group=STRING` для указания родителя создаваемой группы.

5.2 Частичная пересборка артефактов

Для разных типов артефактов предусмотрены отдельные флаги, позволяющие их различать. Так для используемых ресурсов исторически используется флаг `-s`, а для Docker-образов флаг `-d` соответственно.

Изначально парсинг этих флагов был с использованием фреймворка для написания CLI и предполагал задание флагов только с в виде ассоциативного массива, например в виде `resource=12345`. Для поддержания указания ресурсов без их конкретной версии, что актуально для пересборки, парсинг был переписан так:

1. Все флаги парсятся как списки строк.
2. Затем каждая из строк разбивается на две, если есть разделяющий символ `=`. И добавляется в подстановку как есть.
3. Если символа нет, то ресурс добавляется в список на пересборку и пересобирается, если указан флаг `-A`.

Для обеспечения обратной совместимости были написаны юнит тесты на старое и новое поведение, а также проведено ручное тестирование на ресурсах, которые были указаны при обращении пользователей в техническую поддержку. Это обеспечило уверенность в том, что новая функциональность не нарушает обратную совместимость и корректно работает на примерах заинтересованных пользователей.

5.3 Команда `diff` для фермы балансеров

Отношения между контроллерами показаны на рисунке 2.

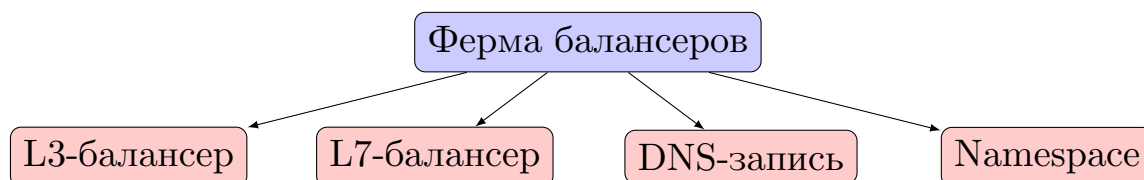


Рис. 2: Отношение между контроллерами

Для каждого из дочерних контроллеров были выполнены следующие действия:

1. Были замоканы все модифицирующие операции в каждом из контроллеров. Все такие операции собираются в общий список событий с предыдущим состоянием ресурса и новым после предполагаемого применения изменений.
2. Все объекты внутри события конвертируются из proto-сообщений в формат yaml.
3. Для каждого события из списка вычисляется локальный diff между текстом с помощью внутреннего инструмента и складывается в список.
4. Полученный список строк конкатенируется и возвращается пользователю.

Основная сложность в работе с такими объектами — определение, какие операции могут произойти в контроллере, и их корректная обработка.

Команда diff для фермы вызывает каждый из своих дочерних объектов независимо, поэтому возникновение ошибки не влияет на результат работы других.

Из-за особенностей работы системы балансировки нагрузки diff между фактически одинаковыми объектами мог различаться. После обсуждения с командой этого сервиса таких случаев, они были исключены из общего результата. Аналогично были исключены поля используемые нашим инструментом для хранения метаданных, так как они не относятся к конечной системе.

После доработок со стороны импорта полученная разница между объектами стала почти нулевой, за исключением случаев явной простановки значений по-умолчанию со стороны infractl.

6 Адаптация

Спустя три месяца после внедрения флага для частичной пересборки артефактов, в чат технической поддержки не поступало вопросов, связанных с использованием команды `build`. За указанный период новая функциональность была использована 74 раза, что составляет 7% от общего числа вызовов команды. Этот показатель демонстрирует востребованность новой опции в рабочих процессах, особенно в сценариях, требующих оптимизации времени сборки.

Новые флаги для создания групп безопасности нашли применение в работе команды, занимающейся развитием внутренней платформы для разработчиков. На их основе был автоматизирован процесс создания нового приложения, что ускорило выполнение типовых задач при развёртывании инфраструктуры и снизило риск ошибок на этапе ручной настройки.

Команде одного из подразделений компании удалось перенести 55 балансеров под управление `infractl` благодаря внедрению команды `diff` для анализа конфигураций фермы балансеров. Это упростило процесс валидации импортированных объектов и минимизировало различия с текущим состоянием конечной системы. Таким образом, пилотная интеграция продемонстрировала готовность инструмента к масштабированию, и в настоящий момент идёт обсуждение следующего этапа внедрения, охватывающего другие команды подразделения.

Заключение

В рамках работы были выполнены следующие задачи:

1. Проанализированы обращения в чат поддержки, открытые задачи и прямые пользовательские запросы.
2. Сформированы требования к новым функциональным компонентам.
3. Реализованы:
 - Возможность частичной пересборки исполняемых артефактов.
 - Флаги для создания групп безопасности в соответствующей системе управления.
 - Команда для построения diff настроек фермы балансировщиков и всех её составных компонентов, за исключением DNS-записей.
4. Проведен анализ влияния изменений на пользователей, показавший:
 - Прекращение обращений в поддержку по частичной пересборке.
 - Востребованность частичной пересборки в рабочих процессах.
 - Активное использование новых флагов для создания новых приложений командой по развитию внутренней платформы для разработчиков.
 - Перенос 55 балансеров одного подразделения компании под управление infractl. И начало второго этапа внедрения платформы в процессы.

В качестве дальнейшего развития работы можно рассмотреть:

- Поддержку DNS-записей в ферме балансеров.
- Реализацию k8s контроллера для декларативного управления группами безопасности.

Код находится под договором о неразглашении и не может быть предоставлен. Отзыв консультанта из компании приложен.

Список литературы

- [1] Crossplane introduction. — URL: <https://docs.crossplane.io/latest/getting-started/introduction/> (дата обращения: 21 декабря 2024 г.).
- [2] Kubernetes basics. — URL: <https://kubernetes.io/docs/tutorials/kubernetes-basics/> (дата обращения: 21 декабря 2024 г.).
- [3] Terraform documentation. — URL: <https://www.terraform.io/use-cases/infrastructure-as-code> (дата обращения: 21 декабря 2024 г.).
- [4] Введение в современную сетевую балансировку и проксирование. — URL: <https://habr.com/ru/companies/vk/articles/347026/> (дата обращения: 21 декабря 2024 г.).
- [5] Документация по группам безопасности Yandex Cloud. — URL: <https://yandex.cloud/ru/docs/vpc/concepts/security-groups> (дата обращения: 21 декабря 2024 г.).