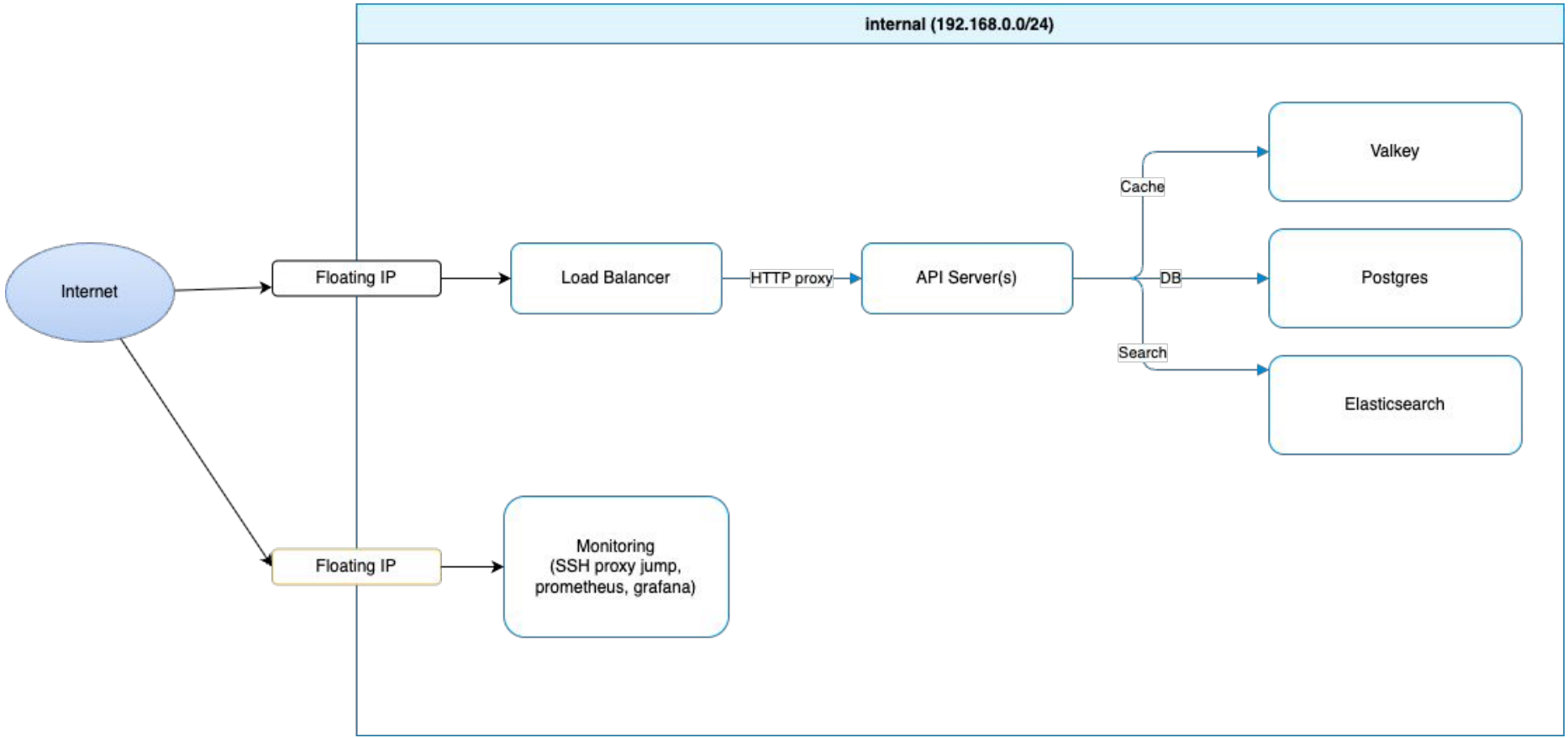
Hackload

Команда - Bulbul

Команда

@skythet	DevOps	Больше 10 лет опыт работы
@FanskiY	Go Developer	Больше 10 лет опыт работы
@serikshaikamalov	Frontend	
@langrenn	(Backend developer)	
Claude Code	Fullstack developer	



Infra readme: https://github.com/hackload-kz/Bulbul/blob/main/infra/README.md

Load balancer	Сервер с nginx	CPU - 4, RAM - 4Gb
API servers	Наш сервис, написан на Golang	CPU - 4, RAM - 4Gb
Valkey	Форк редиса, для кэширования	CPU - 1, RAM - 2Gb
Postgres	База данных	CPU - 4, RAM - 8Gb
Elasticsearch	Используется для fulltext search events	CPU - 8, RAM - 16Gb
Monitoring	Для мониторинга серверов, а также в качестве SSH proxy jump сервера	CPU - 4, RAM - 4Gb

Поиск событий

/api/events?page=1&pageSize=20

Про кэшинг, профайлинг и настроек nginx

Load balancer (nginx)

1. Быстро выяснили что 1 CPU, 1Gb RAM слишком мало для load balancer. Но это еще зависит как хорошо работает ваш сервис.
2. Ошибки **“Cannot assign requested address”** на nginx “errors.log”. Настройли **“net.ipv4.ip_local_port_range = 1024 65535”**
3. Пример настроек upstream:

```
upstream backend {
    least_conn;
    server 192.168.0.52:8081 max_fails=0;
    server 192.168.0.219:8081 max_fails=0;
    keepalive 256;
}
```

Кэширование

1. Сделали кэшинг для запросов где в фильтрах только page & pageSize и pigeSize кратно к 5;
2. Запустили pprof (https://pkg.go.dev/net/http/pprof), увидели что json.Unmarshal забирает большое количество времени: мы читали events из кэш но делал decoding на go структуру а дальше Gin (web framework) обратно делал json encoding;
3. Включили client side caching: https://valkey.io/topics/client-side-caching/

Архивные события

/api/events?query=***&date=***

Поиск лучшего fulltext search инструмента

Postgres fulltext search

1. Создали fulltext index для events: https://github.com/hackload-kz/Bulbul/blob/1ec45615839ef92134ab796cb7508177e4a1dd20/infra/roles/postgresql/templates/fulltext-index.sql
2. Поиск:

```
SELECT id, title, description, ts_rank_cd(search_vector, query) AS rank
FROM events_archive, to_tsquery('russian', 'Концерт') query
WHERE search_vector @@ query
ORDER BY rank DESC;
```

Mongodb fulltext search

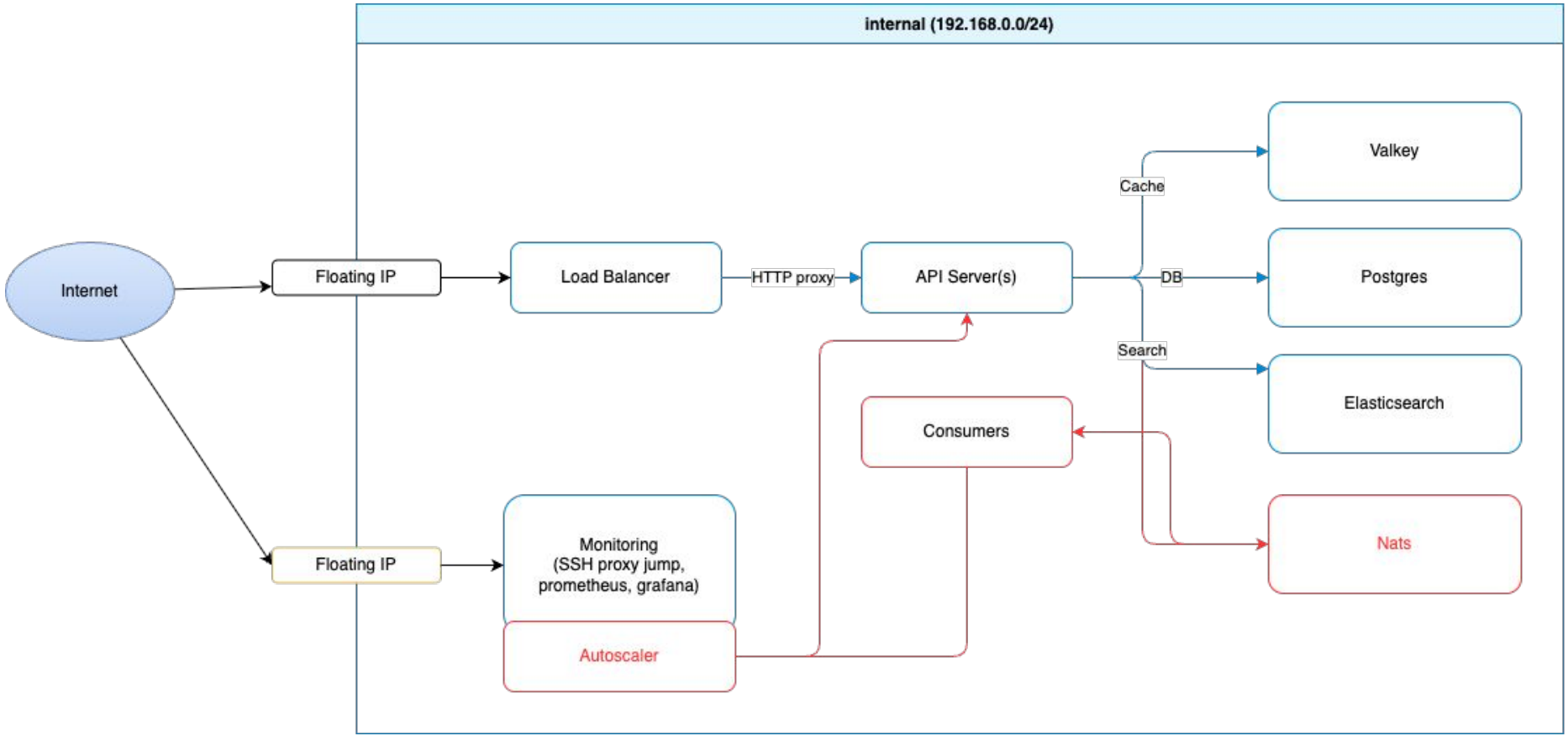
Создание индекса	events_collection.create_index([("title", TEXT), ("description", TEXT)], default_language="russian", weights={"title": 10, "description": 5}, name="events_text_search_russian")
Поиск	db.events_archive.find({ $text: { $search: “Концерт } })
Попробовали использовать Read Preference	https://www.mongodb.com/docs/manual/core/read-preference/

В итоге: Elasticsearch

1. В итоге пришли к Elasticsearch, на машине с 8 CPU и 16Gb памяти прошли тесты для 1000 и 5000 пользователей
2. Потом сделали чтобы elasticsearch был основным источником “events”, их убрали из postgres

Разное

1. IaC (terraform + ansible)
    a. Позволило экономит
    b. Позволило экспериментировать
2. Рекомендую попробовать mise: https://github.com/jdx/mise
3. Claude Code
    a. Было бы невозможно успеть и попробовать все что мы попробовали без этого
    b. Может создать ansible roles, которых можно просто подкорректировать чтобы начать использовать
    c. Может генерировать grafana dashboards, просто дайте какие у вас есть метрики и какой layout хотите
    d. Ревью результата очень важен, claude от себя может добавить кода который сильно может повлиять на результаты


internal (192.168.0.0/24)
Internet
Floating IP
Floating IP
Load Balancer
HTTP proxy
API Server(s)
Cache
DB
Search
Valkey
Postgres
Elasticsearch
Nats
Consumers
Monitoring
(SSH proxy jump,
prometheus, grafana)
Autoscaler

Спасибо к организаторам!

Было интересно и познавательно