CONCIERGE

TEAM 6 TECHNIQUE SHARING PRESENTATION

RAINCLOUDED LEEROY JOSH MYKOLA NATHAN

What are we discussing today?

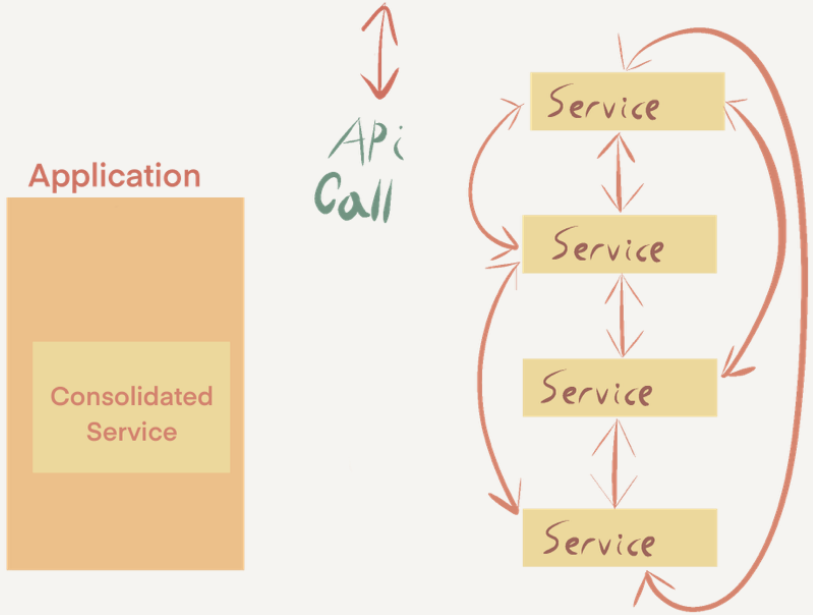
Microservices

What is the motivation?

To enforce separation of concerns as the project grows, there is a requirement to decide how to keep modules decoupled.

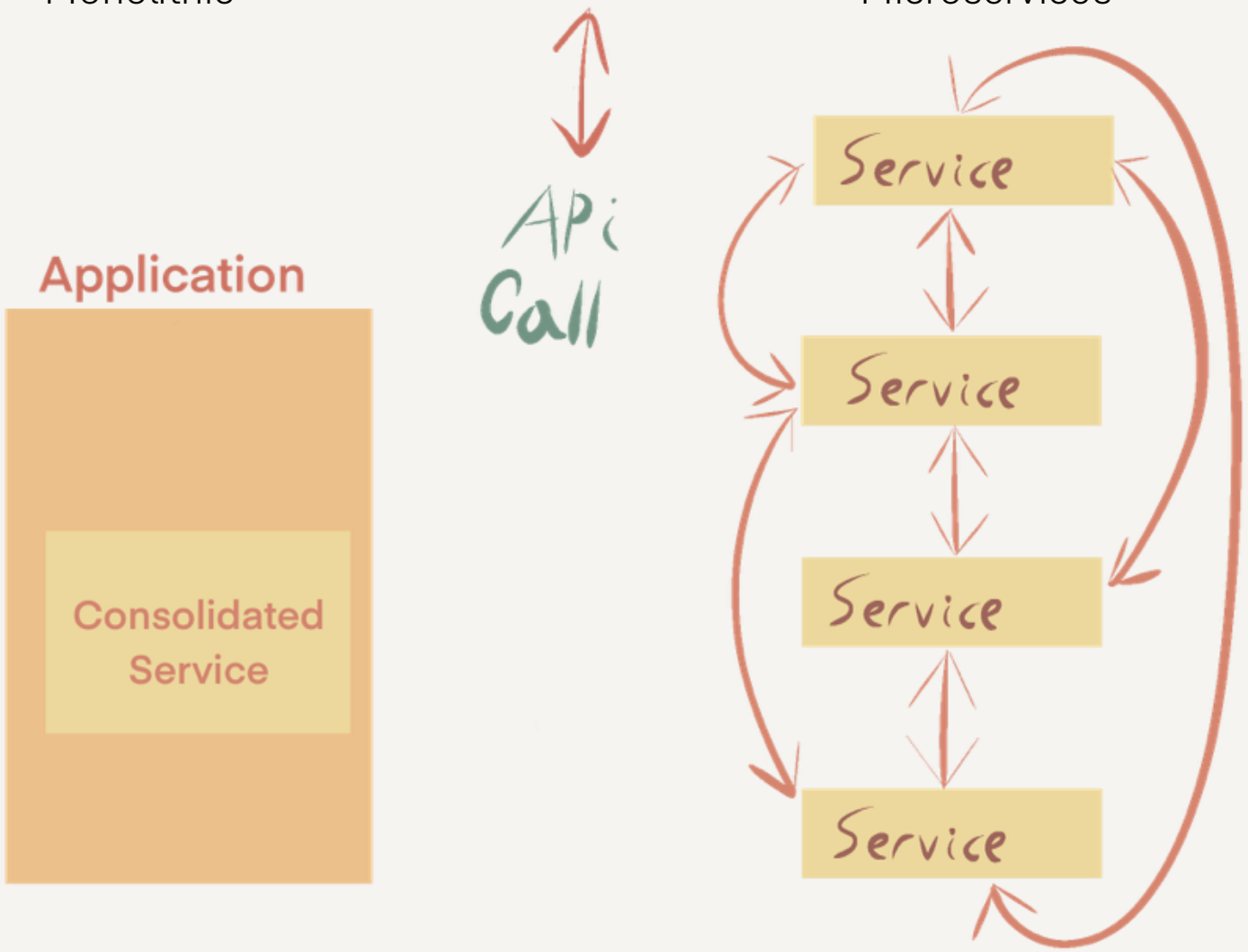
First things first
What are microservices? What are the alternatives?

- Microservices is an architecture design technique, wherein discrete functional constituents, characterized by being loosely coupled are separated into independent services each including their own development stack engaging with each other utilizing api calls [1,2,3]
- The main alternative is a monolith design. That is where all services are tightly coupled in one large package [1,3]

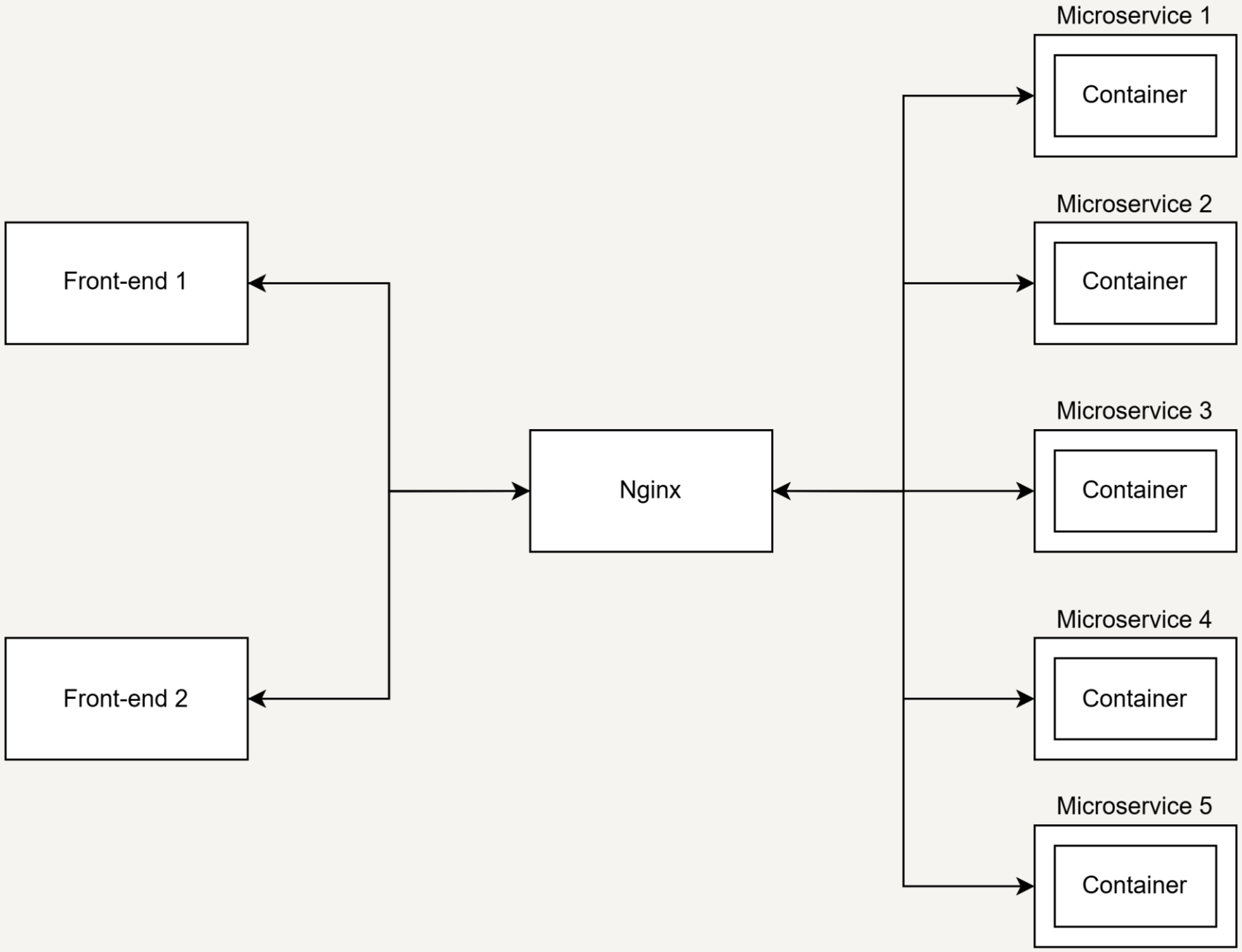


Monolithic

Microservices

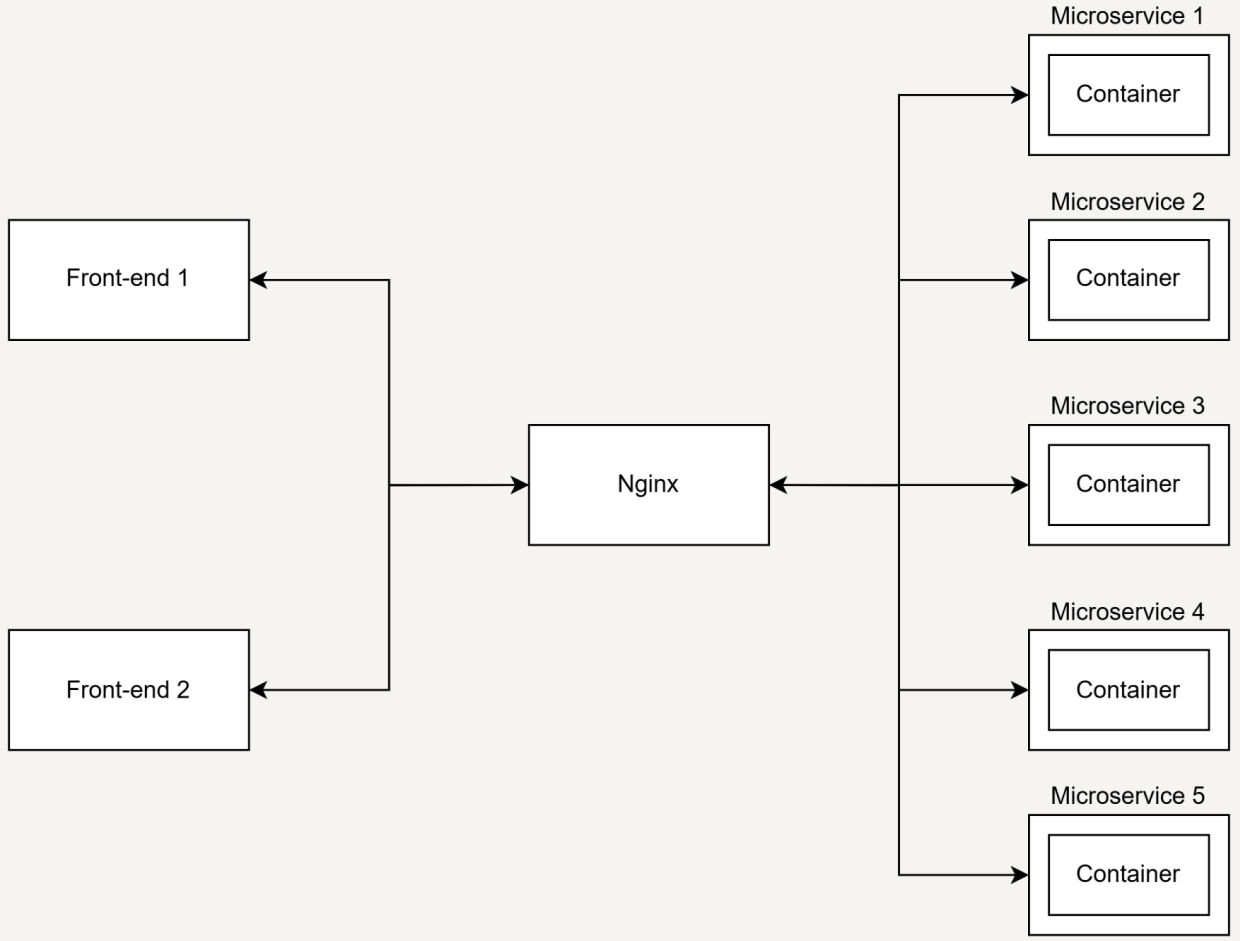


Very Demure


Microservice 1
Container
Microservice 2
Container
Front-end 1
Nginx
Microservice 3
Container
Microservice 4
Container
Front-end 2
Microservice 5
Container

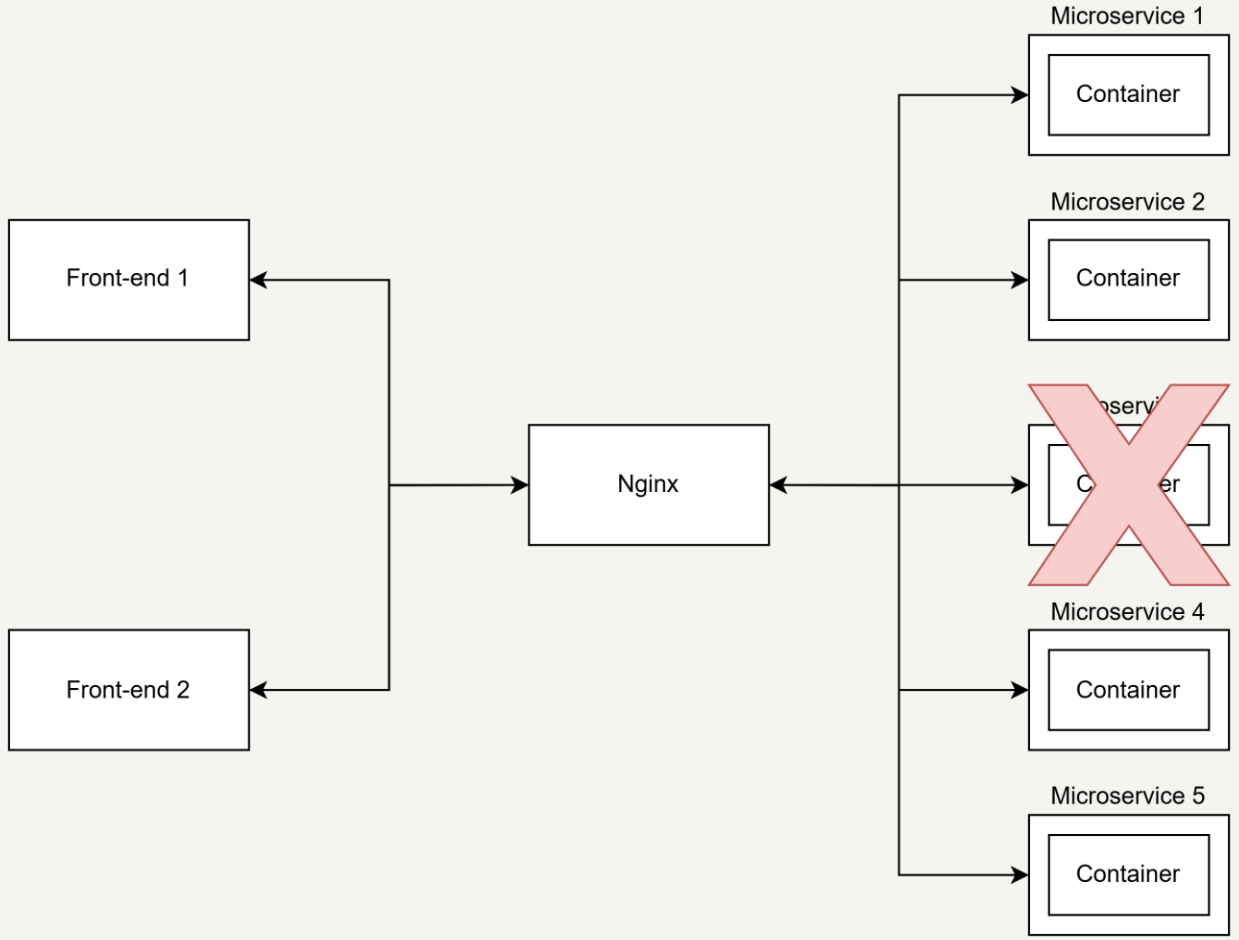
Advantages in Industry

- Why would a company choose a microservice architecture?



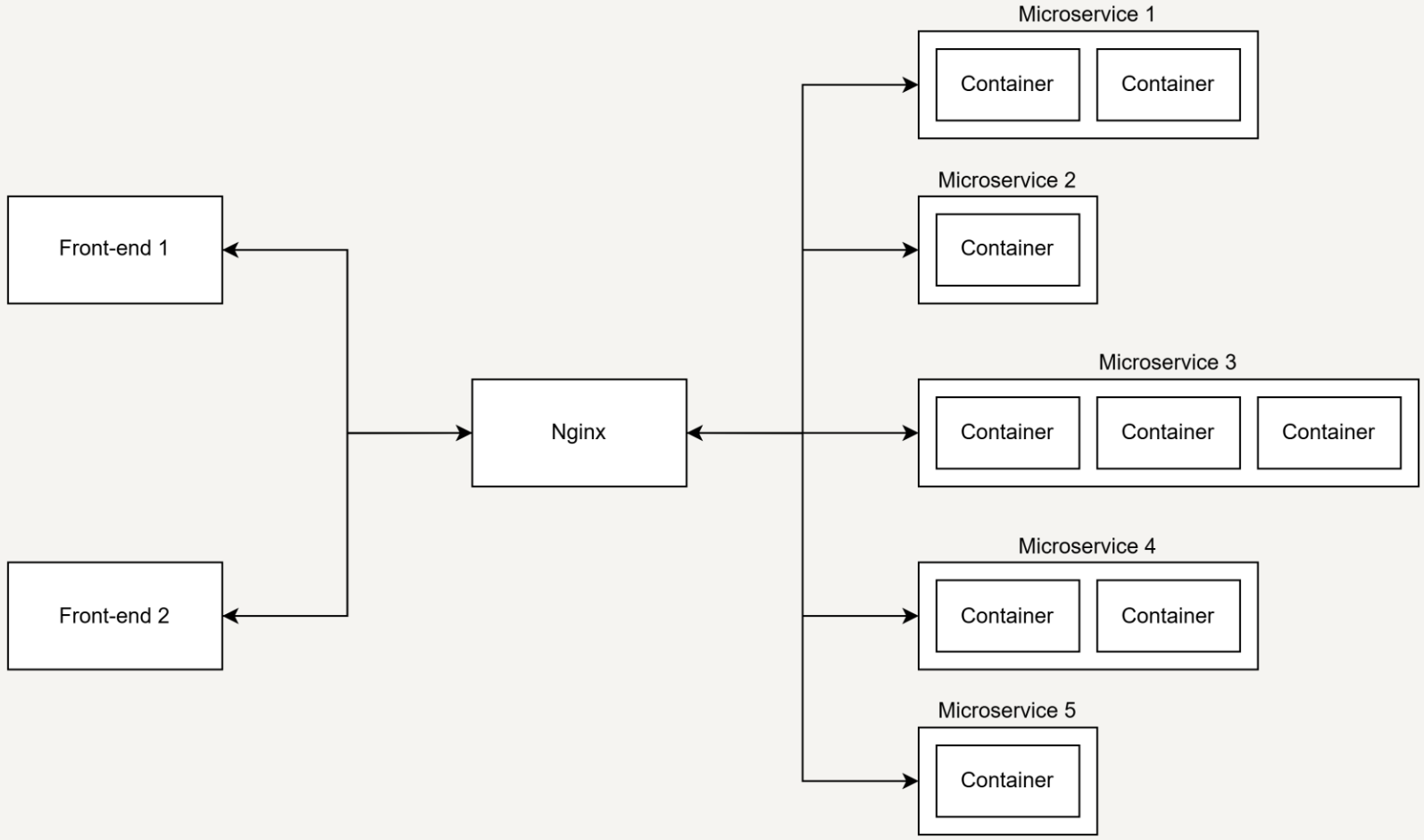
Enhanced Fault Tolerance

- Reduced risk of downed hardware/software affecting other parts of the system



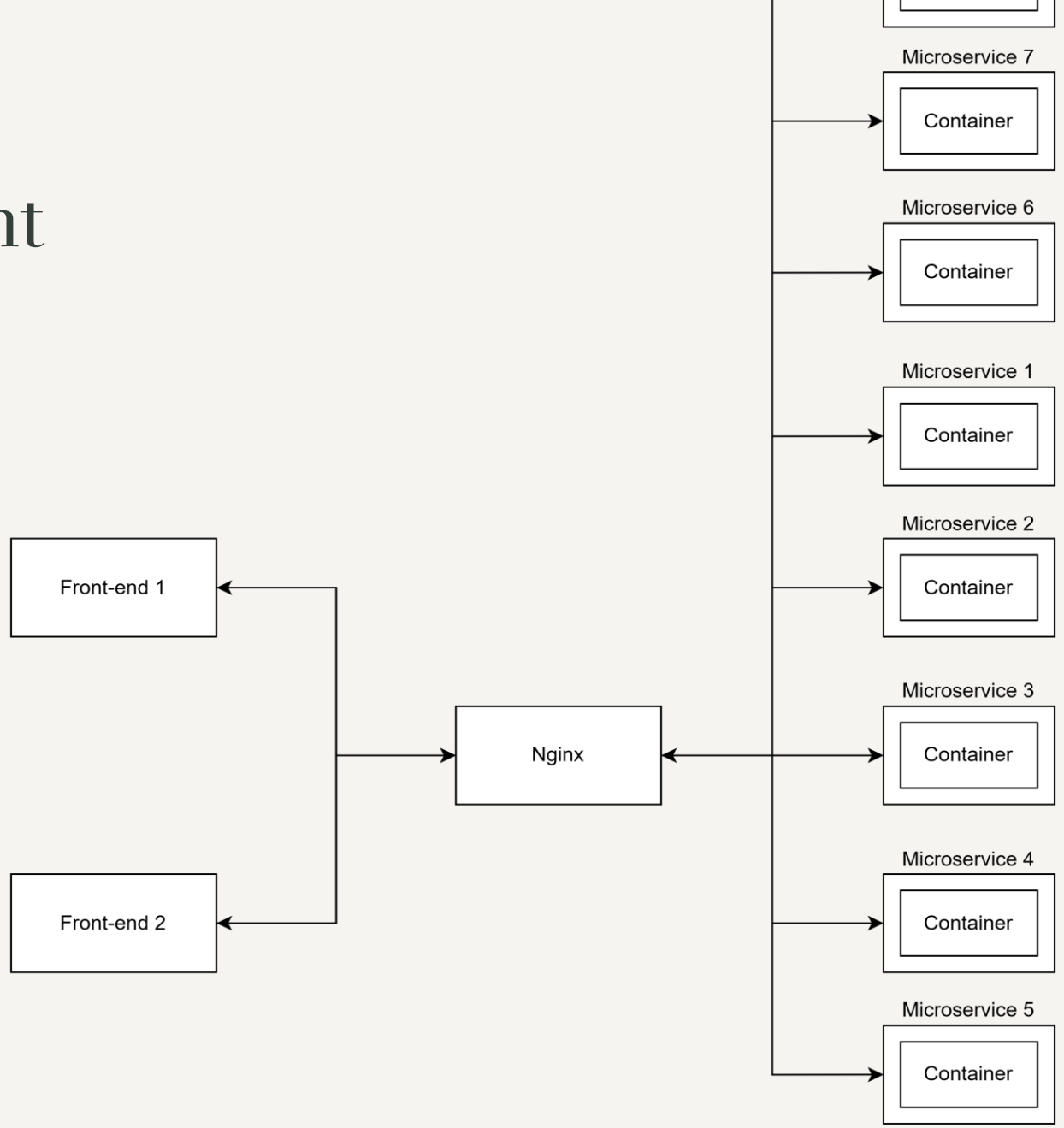
Precise Scaling

- Scale a service individually based on its needs, allowing resources/money to be used more efficiently.



Ease of Deployment

- Each microservice can be deployed on their own.
- Makes updates, fixes, and adding new features faster and with less risk to the rest of the system.



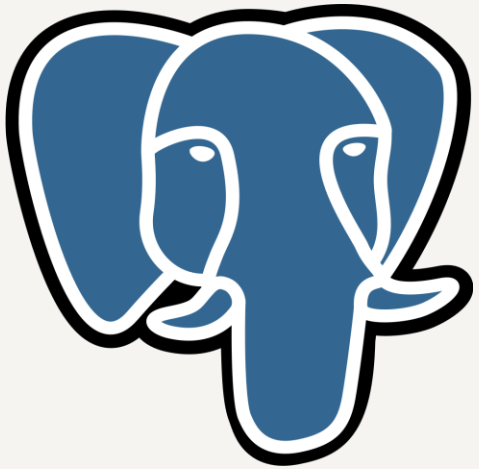
Advantages in our Project

Advantages in our Project

- More opportunities to learn and experiment
  - As microservices were isolated from each other, members implementing a service had the choice of choosing their preferred languages and databases

Advantages in our Project

- More opportunities and experiment
  - Current Architecture
    - Microservices
      - **1 Microservice:** Written in Go
      - **2 Microservices:** Written in Python with Flask
      - **2 Microservices:** Written in C# with .NET
    - Databases
      - **1 Database:** MariaDB
      - **2 Databases:** MongoDB
      - **2 Databases:** PostgresSQL



Advantages in our Project

- More opportunities and experiment
  - Learning to work with these tools and discovering their strengths not only provided us with experience, but as well as insights into what we'd might like to use in future projects.

Advantages in our Project

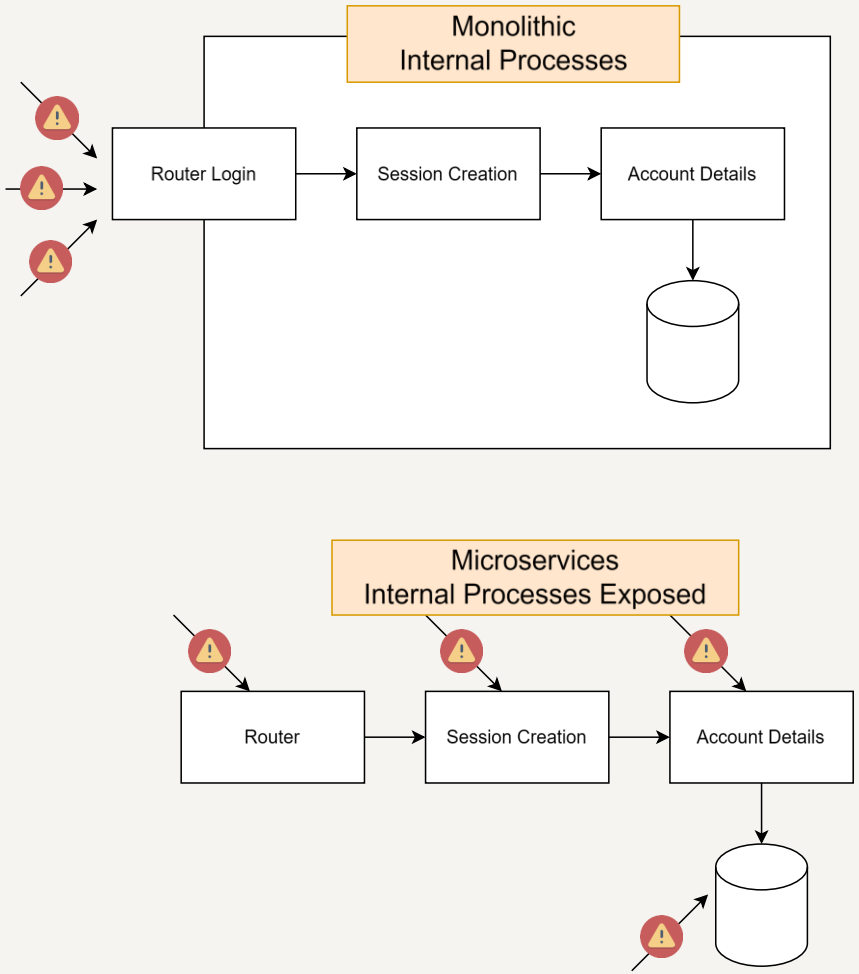
- Smaller bugs
  - As microservices were isolated from each other, bugs had a reduced risk of affecting large parts of the system.
  - This meant that upon visual inspection, it was easier to locate a bug as it was highly likely to be contained in one microservice.
    - Less files and lines of code to search.

  - **Note**: Bugs increase in severity when communication between microservices increase, as you are now debugging within an internal network

Advantages in our Project

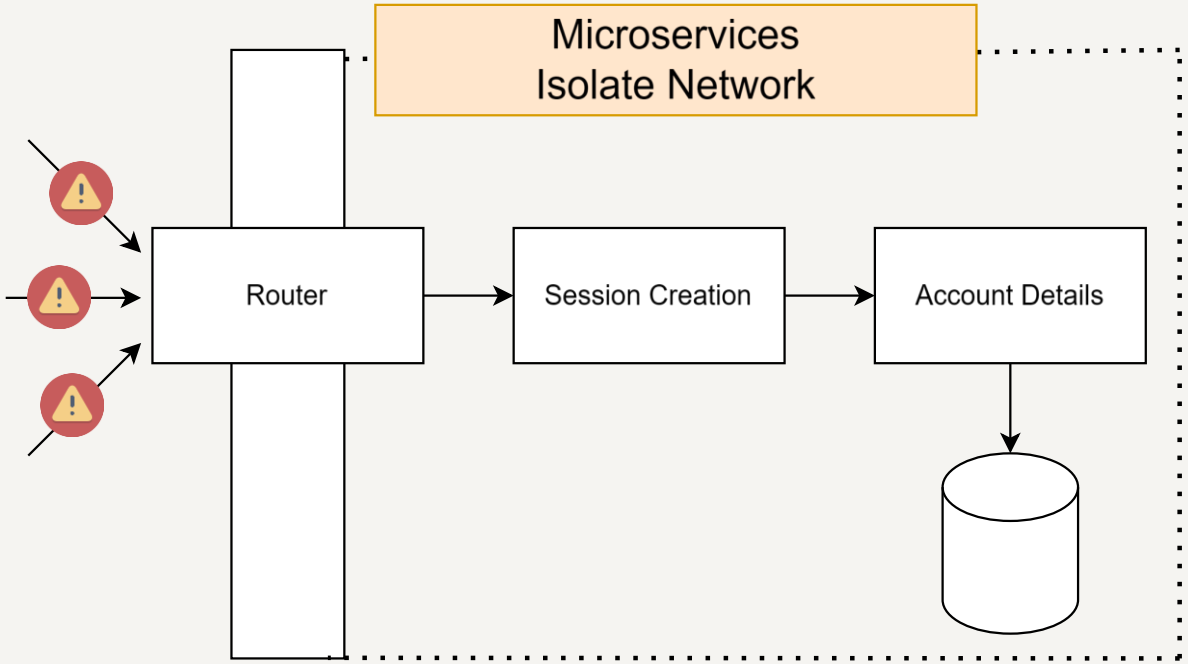
- Faster Development
  - While creating our back-end, members were able to work in parallel when developing their own microservices, which reducede delays and helped us finish things faster.
  - Reduced errors and bugs caused from miscommunication
  - Bad/messy code was isolated in its own server, if it works it works!

Considerations & Challenges



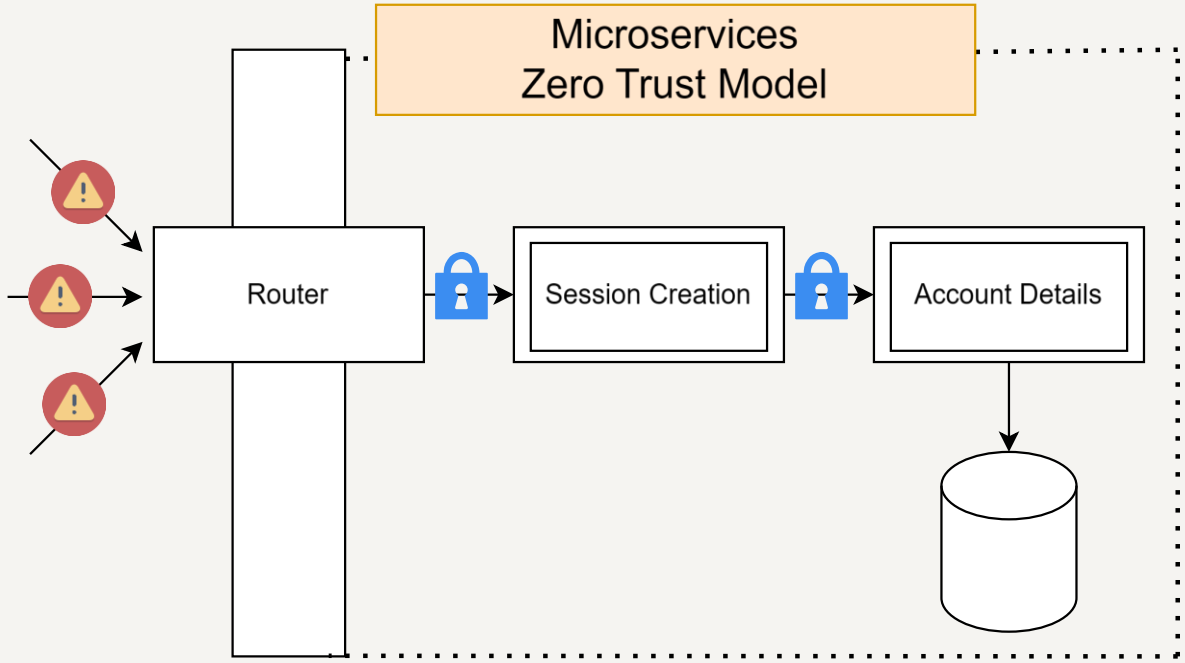
Security & Exposure

Considerations & Challenges



Security & Exposure

Considerations & Challenges



Security & Exposure

Considerations & Challenges

Network Overhead

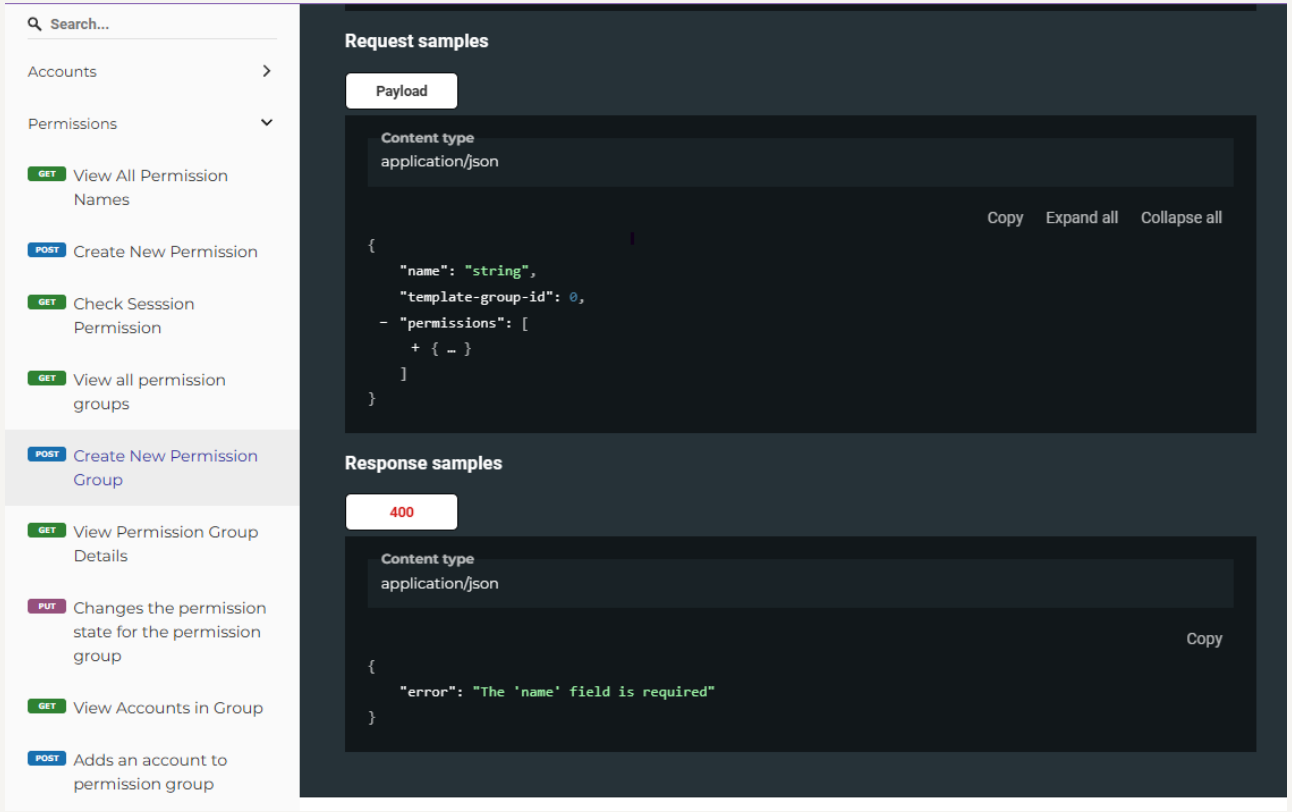
Considerations & Challenges

Network Overhead

Considerations & Challenges

Network Overhead

Considerations & Challenges



API Complexity

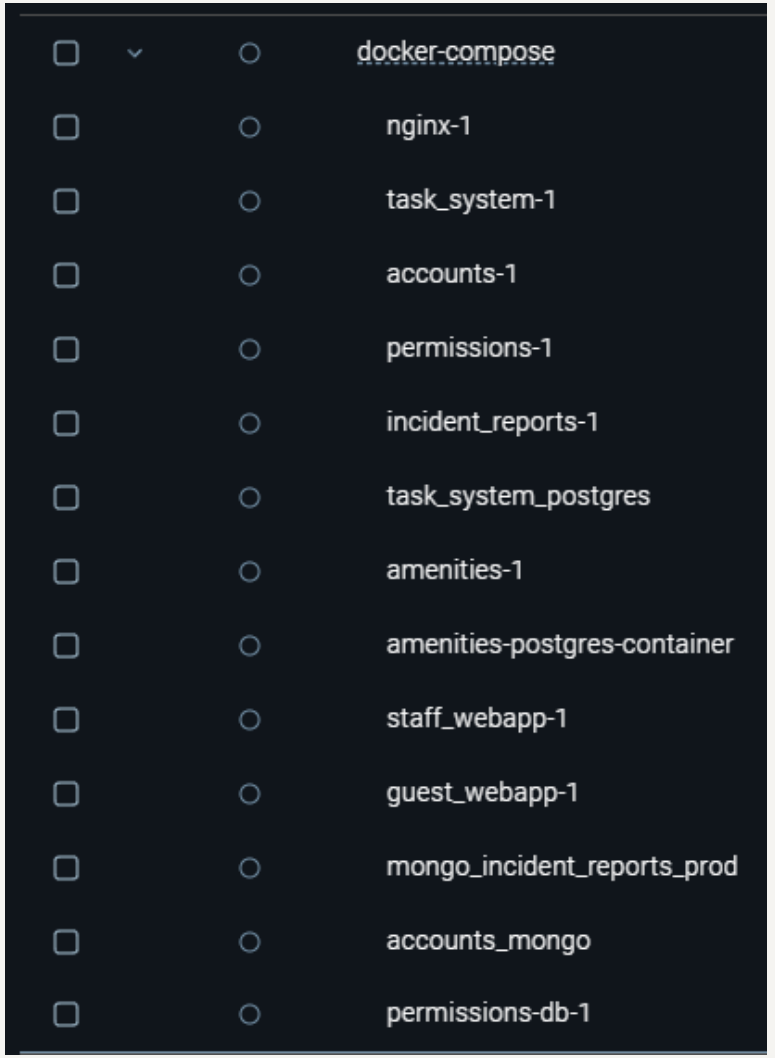
Considerations & Challenges

Documentation
Logging

*Image from the Prometheus Website

API Complexity

Considerations & Challenges



Deploying Many Services

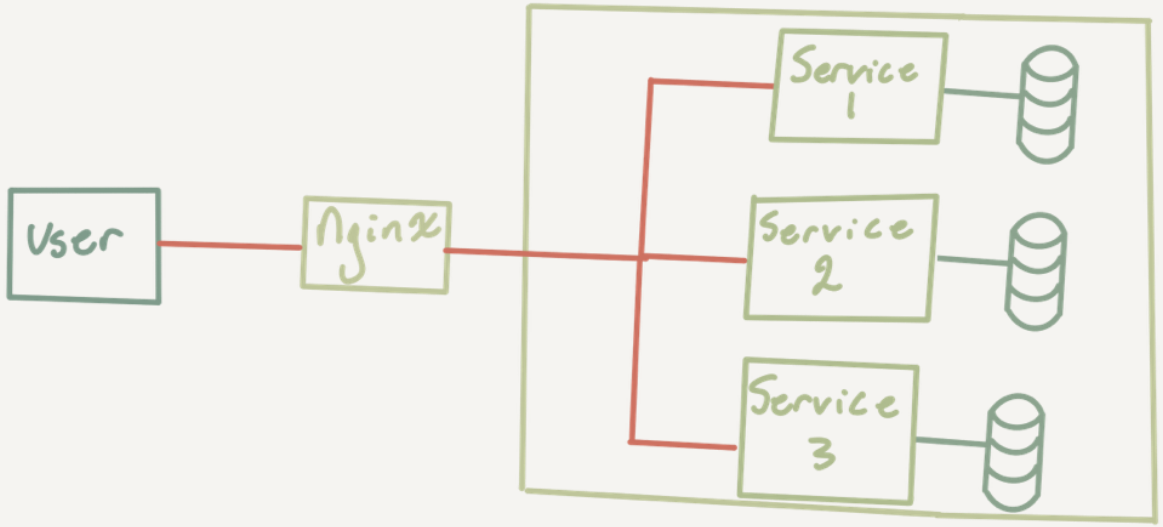
Considerations & Challenges

Testing Across languages

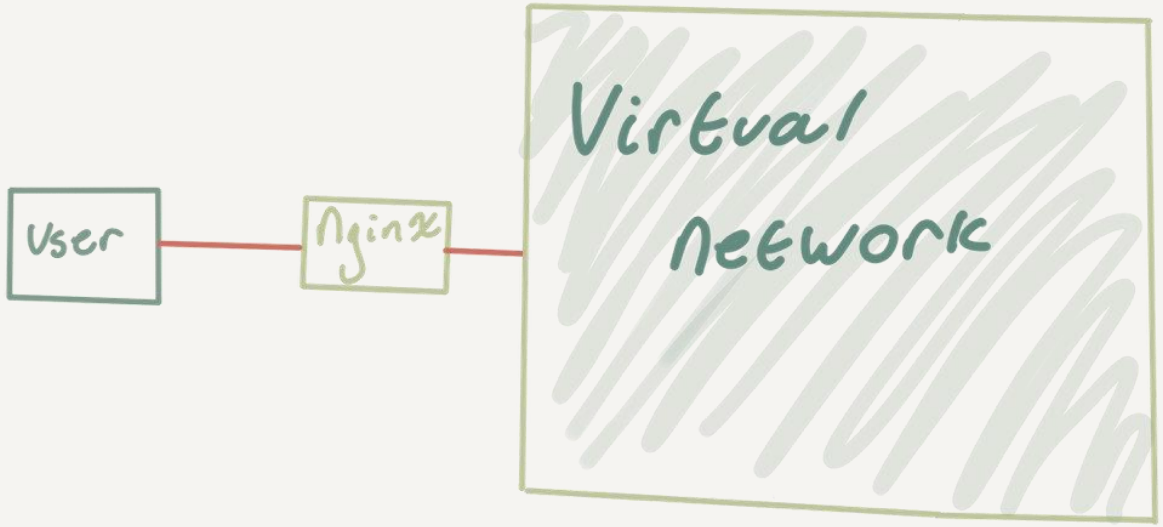
How to run all tests:

1. Accounts
   - `python3 -m pip install src/accounts/requirements.txt`
   - `python3 -m src/accounts/tests`
2. Amenities
   - `dotnet dotnet test --working-directory ./src/amenities/amenities_db_integration_test`
   - `dotnet dotnet test --working-directory ./src/amenities/amenities_test`
3. Guest Webapp
   - `npx cypress run --project ./src/guest_webapp`
4. Incident Reports
   - `python -m unittest discover -s incident_reports_tests -p "*.py" -v`
5. Permissions
   - `go test ./src/permissions/tests`
6. Staff Webapp
   - `npx cypress run --project ./src/staff_webapp`
7. Task System
   - `dotnet dotnet test --working-directory ./src/task_system/task_system_test`

Key Interesting Technique Take-away:



Key Interesting Technique Take-away:



Questions?

Support Us!