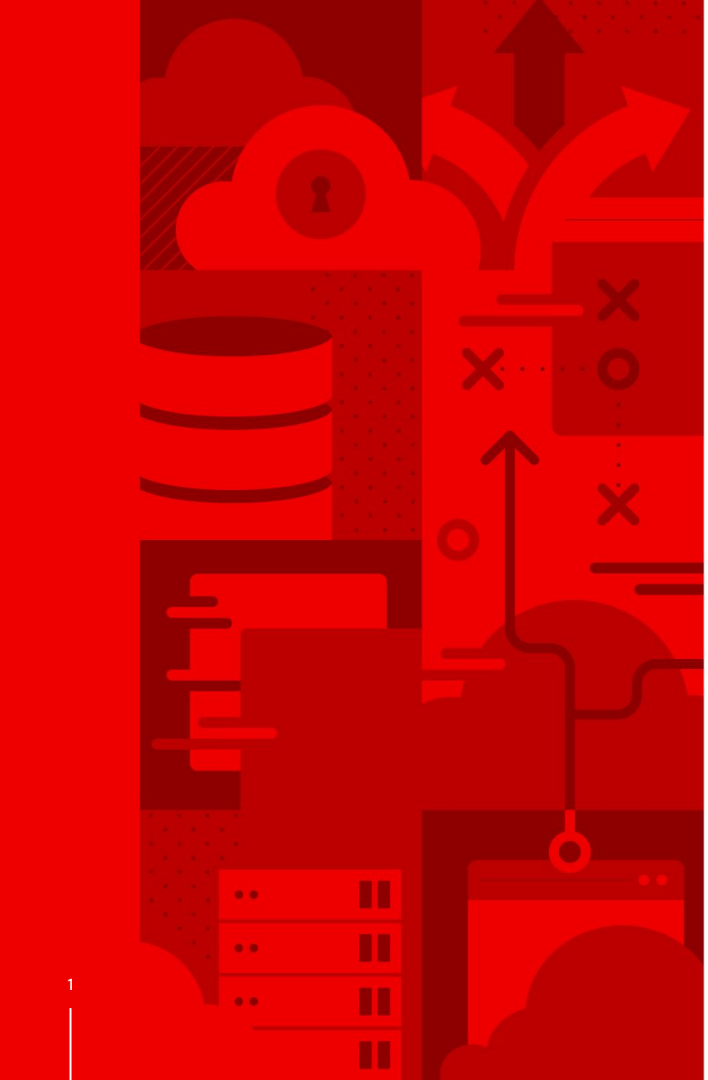


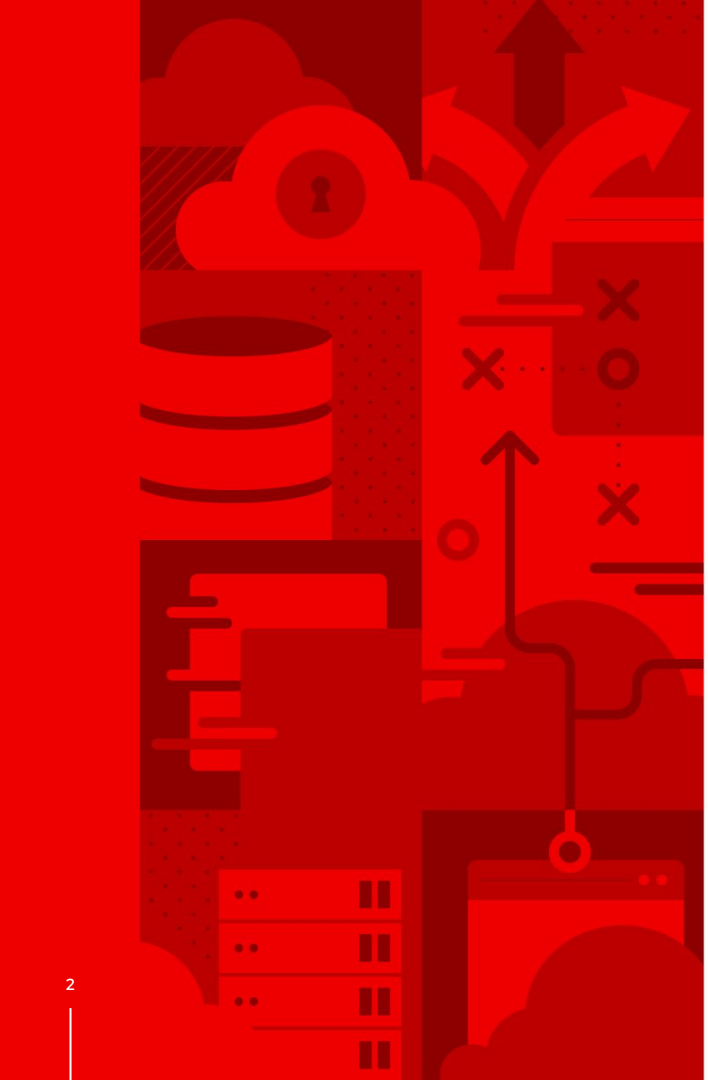
# Clean architecture

## *@Infrastructure level*

Maarten Vandeperre

SSA (AppDev)

Red Hat



# Keep your options open

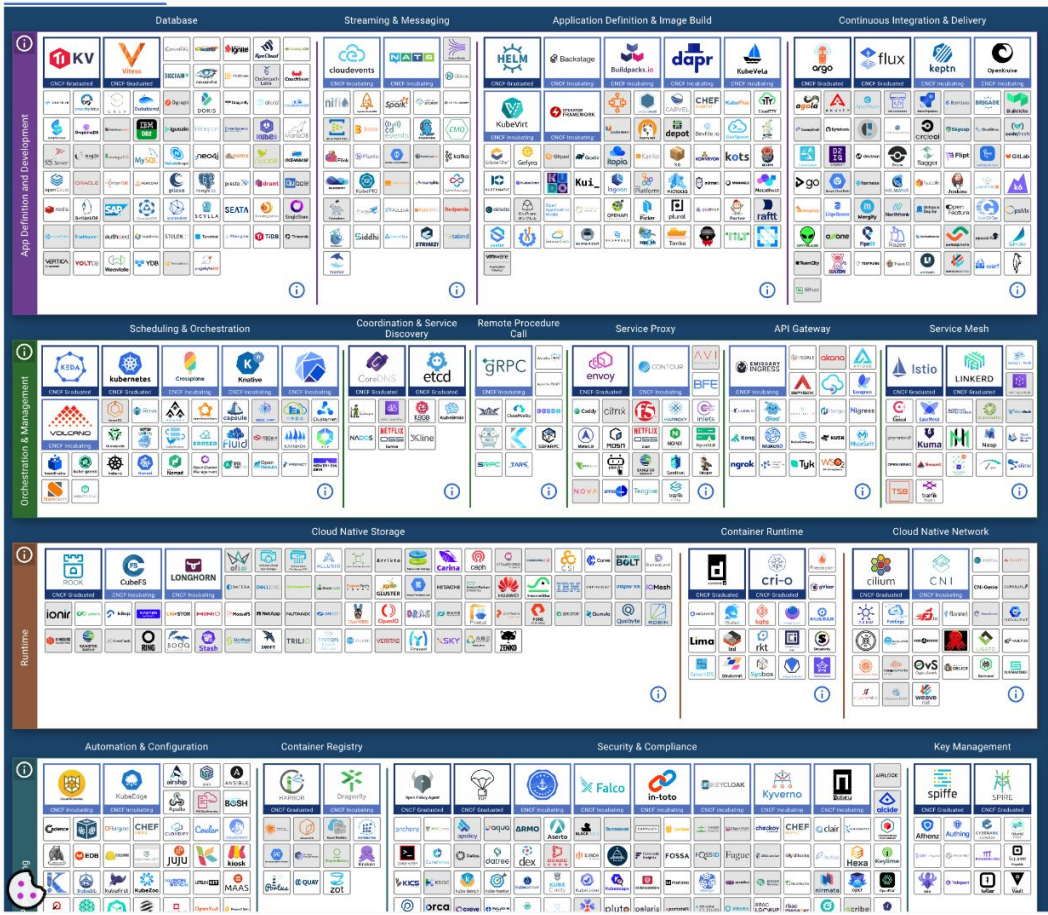
Maarten Vandeperre
SSA (AppDev)

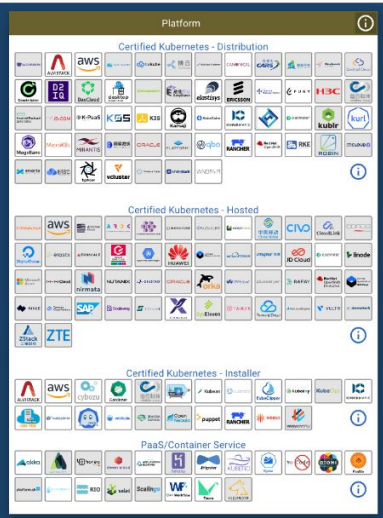


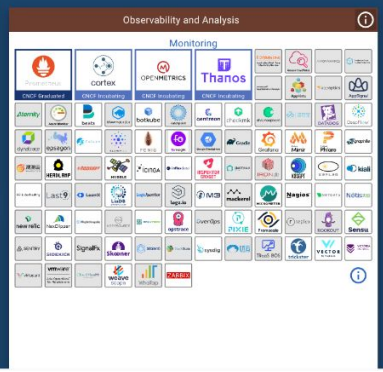
Red Hat

Landscape | Card Mode | Members | Serverless | Wasm

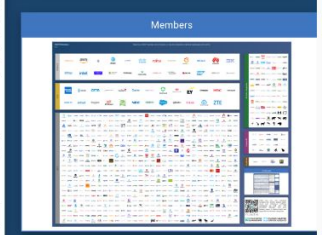
80%











Red Hat


Possibilities
Possibility
Possibility
Possibility
Possibility
Possibility



Red Hat

- + 10 year experience in software development.
- Software architect - technical lead - development.
- @ Red Hat since 01/01/2023.
- Freak about clean architecture.

"DevOps is an illusion"



Red Hat

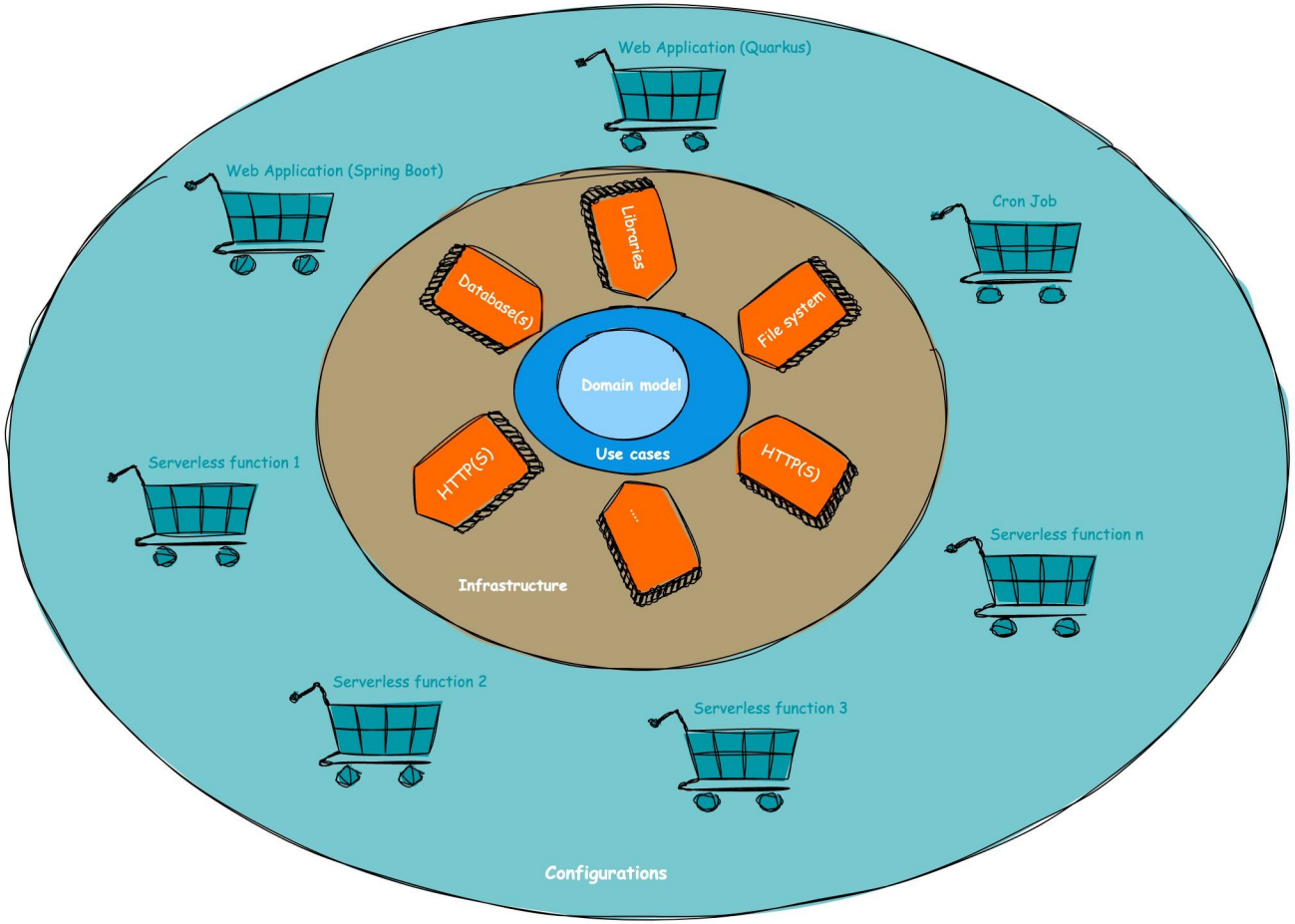
# Agenda

- Clean architecture - concepts
- How to map it on the infrastructure
- Extra's

# Clean architecture

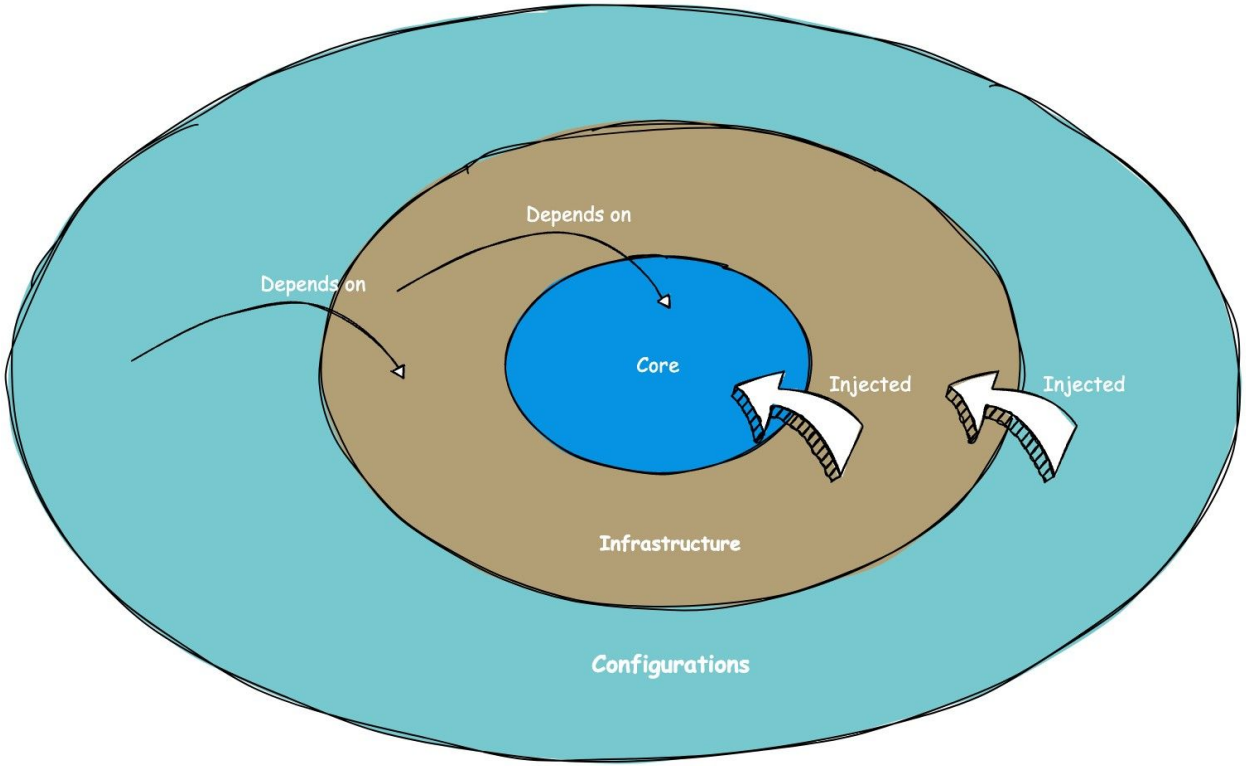
# –

# concepts



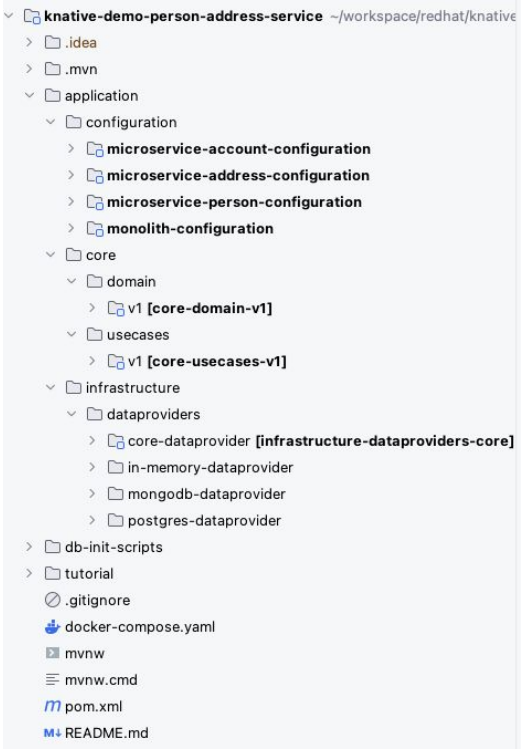

## Clean architecture

- Keep your options open.
- https://developers.redhat.com/articles/2023/04/17/my-advice-building-maintainable-clean-architecture#
- https://developers.redhat.com/articles/2023/04/17/my-advice-transitioning-clean-architecture-platform
- OpenShift as core.
- Knative as core.
- Keep on using cloud services.




## Clean architecture

- Keep your options open.
- https://developers.redhat.com/articles/2023/04/17/my-advice-building-maintainable-clean-architecture#
- https://developers.redhat.com/articles/2023/04/17/my-advice-transitioning-clean-architecture-platform
- OpenShift as core.
- Knative as core.
- Keep on using cloud services.




```
class DefaultCreatePersonUseCase(
    private val personRepository: PersonRepository
) : CreatePersonUseCase {
    maartenvandeperre +1
    override fun execute(requestData: CreatePersonUseCase.Request): CreatePersonUseCase.Response {
        if (requestData.firstName == null) {
            throw ValidationException("First name should not be null")
        }
        if (requestData.lastName == null) {
            throw ValidationException("Last name should not be null")
        }
        if(requestData.addressRef != null){
            try {
                UUID.fromString(requestData.addressRef)
            } catch (e: Exception) {
                throw ValidationException("Address ref is not a UUID format")
            }
        }
        return CreatePersonUseCase.Response(
            personRepository.save(
                PersonRepository.DbPerson(
                    ref = UUID.randomUUID(),
                    firstName = requestData.firstName,
                    lastName = requestData.lastName,
                    birthDate = requestData.birthDate,
                    addressRef = requestData.addressRef?.let { UUID.fromString(it) }
                )
            )
        )
    }
```

```
<profile>
  <id>microservice-account</id>
  <modules>
    <module>application/configuration/microservice-account-configuration</module>
    <module>application/core/domain/v1</module>
    <module>application/core/usecases/v1</module>
    <module>application/infrastructure/dataproviders/core-dataprovider</module>
    <module>application/infrastructure/dataproviders/in-memory-dataprovider/v1</module>
    <module>application/infrastructure/dataproviders/postgres-dataprovider/v1</module>
    <module>application/infrastructure/dataproviders/mongodb-dataprovider/v1</module>
  </modules>
</profile>
```

## Clean architecture

- Enforce by compilation.
- Easy to plug-and-play.
- Use cases instead of SOA.
- No DRY.
- Isolation of logic.
- Withstand the test of time.
- https://github.com/maarten-vandeperre/clean-architecture-software-sample-project


Possibilities
Possibility
Possibility
Possibility
Possibility
Possibility

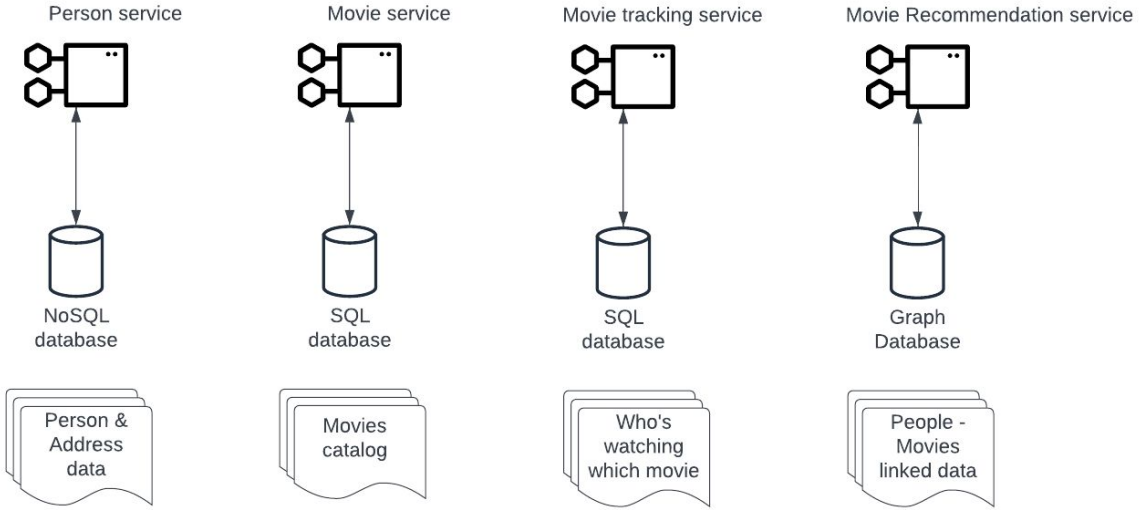
# Clean architecture

# –

# Map it on the infrastructure level

# Why hybrid- and/or multi-cloud?

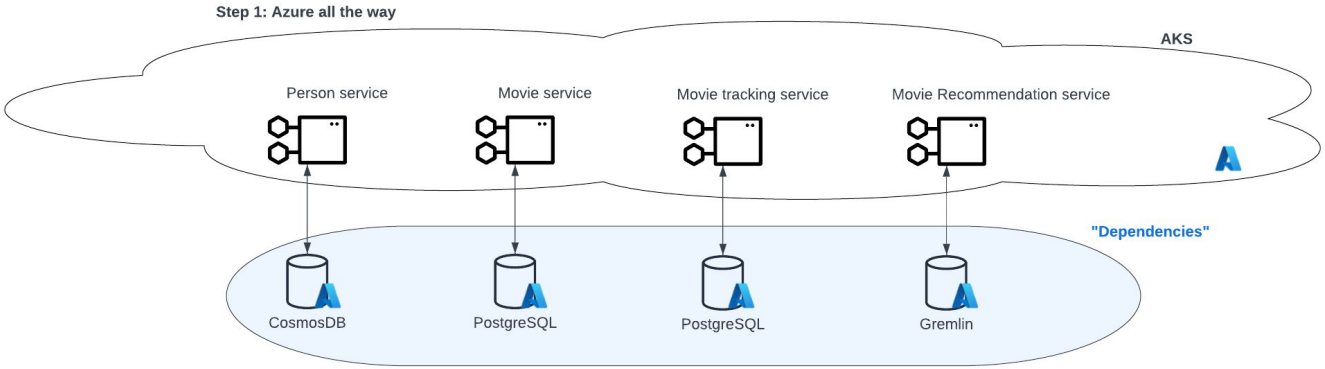
**First phase: the basic architecture**




## Demo architecture outline

- Person service
- Movies catalog service
- Movie tracking service
- Movie recommendation service

# Clean architecture @ infrastructure level



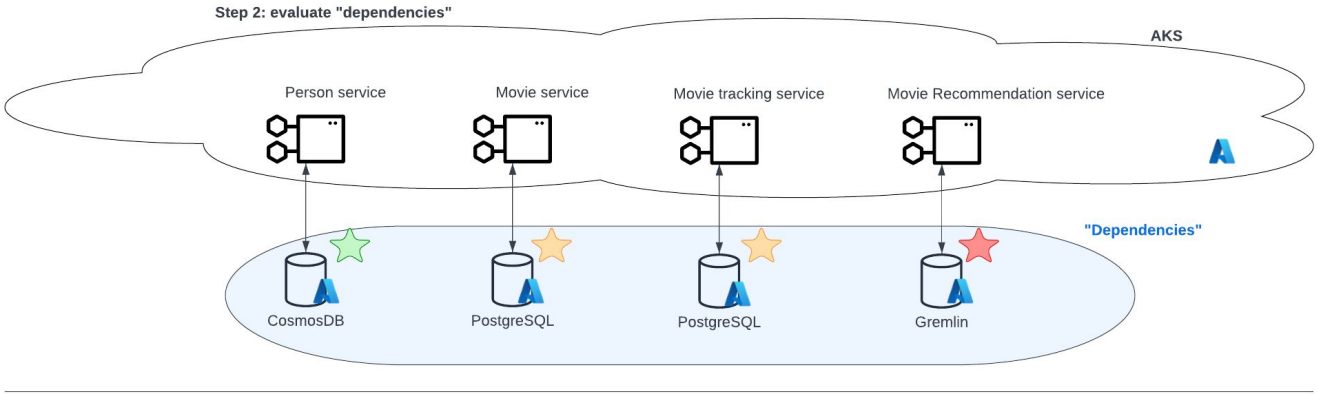

## First platform design

- All Azure
- NoSQL database ⇒ CosmosDB
- SQL database ⇒ PostgreSQL
- Graph database ⇒ Gremlin
- DIY ⇒ AKS





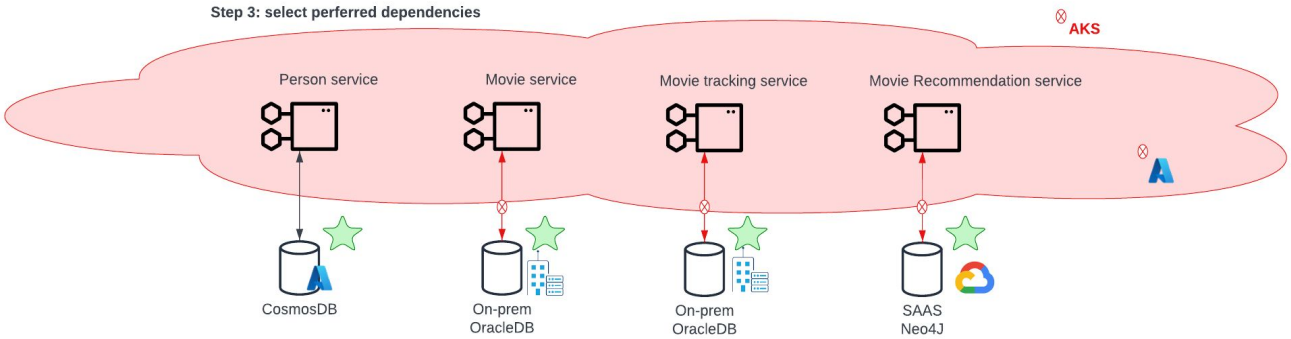
Red Hat




## Evaluation of the dependencies

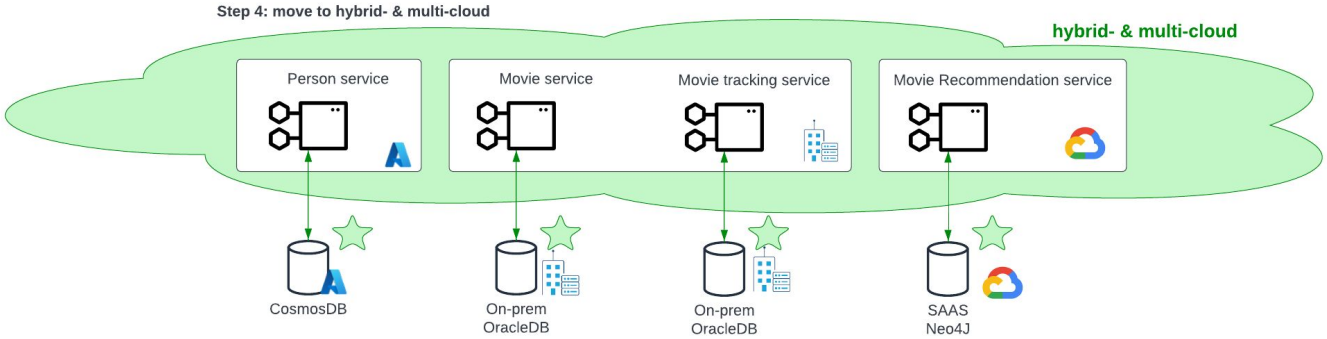
- CosmosDB ⇒ accepted
- PostgreSQL ⇒ On-premise OracleDB is preferred
- Gremlin ⇒ no-go
- AKS ⇒ seems cheaper ⇒ accepted

!!! Risks and hidden costs that go along with a DIY solution were overlooked during the evaluation.




## Updated platform design

- All dependencies accepted
- Dependencies not available within the “programming language” AKS or Azure
- Analogy with Java & Python dependencies ⇒ GraalVM
- Search GraalVM solution for AKS/Azure

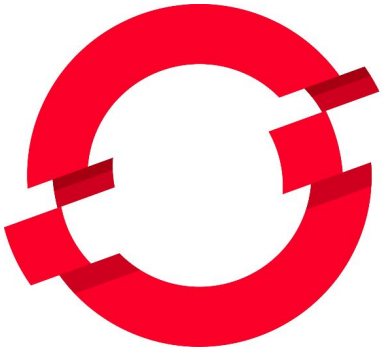



## "Programming language" solution

- ***Keep your options open***
- GraalVM analogy ⇒ go hybrid
- Hybrid- & multi-cloud

⇒ The platform will be more resilient, will require less maintenance (costs and time) and allows (fairly easily) to be innovated in the future.

⇒ Platform is open for innovation, can grow/transform with the organization.



RED HAT®
OPENSHIFT
Container Platform

# Can bring clean architecture to the infrastructure level

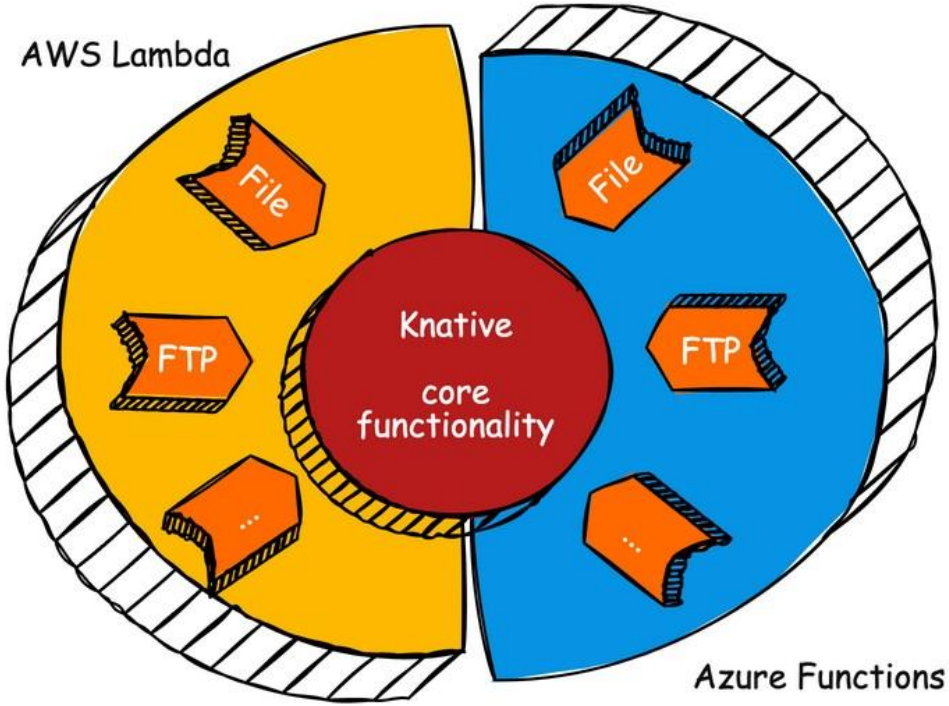
## Disclaimer

Although we used OpenShift as a solution for the issues that come along with EKS, OpenShift is way more than just a Kubernetes installation, it's a full application/container platform.

# Extra's on the infra side

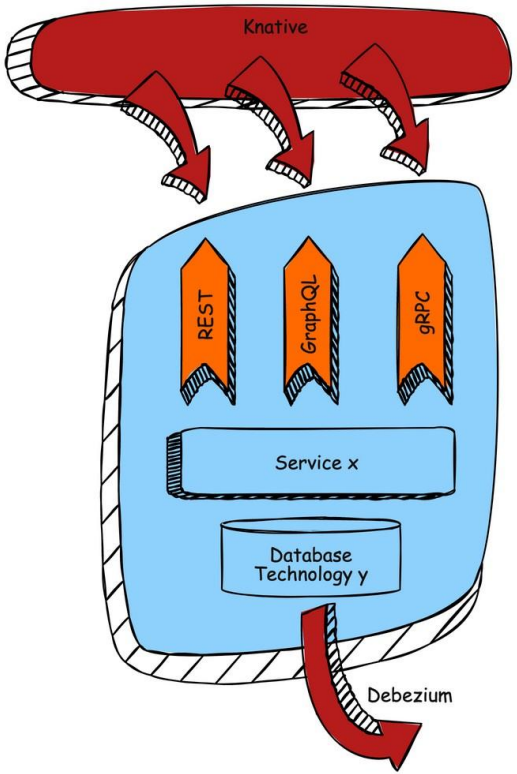
# –

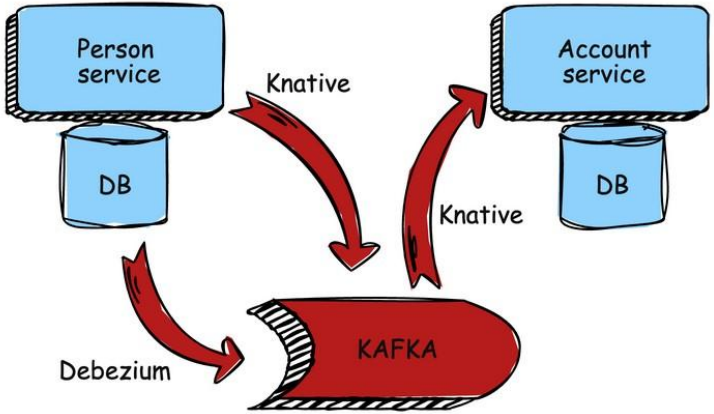
# Use cases




## Clean architecture

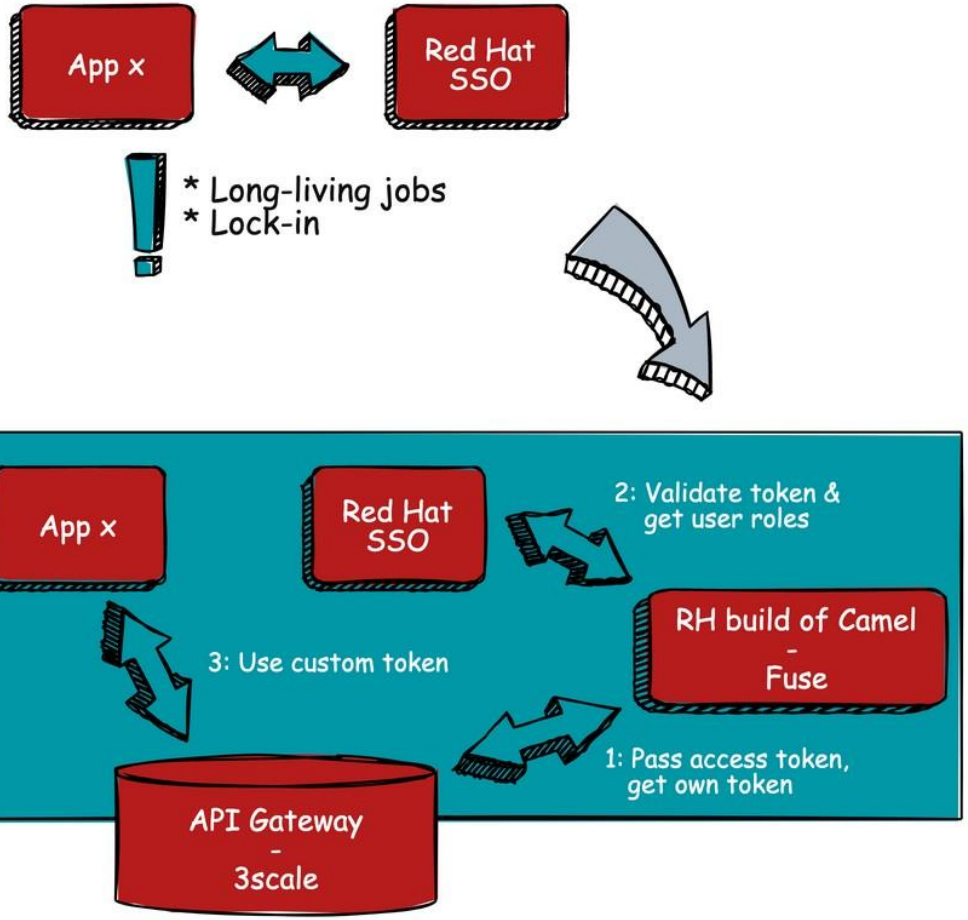
- Serverless: no competition between Knative and Lamda/Functions
  ⇒ play on different levels.
- Knative: core layer.
- Lambda/Function: infrastructure layer.


## Clean architecture

- Standardization, without hijacking innovation.
- Standardized tech stack, but open for other tooling.




## Clean architecture

- Abstract away issues.
- Infrastructure (e.g., Camel) as interface.
- Use the toolbox.

  ⇒ Developer experience.

# Thanks

## &...

Keep your code and architectures clean ;)



Red Hat