



KNEE DEEP IN MICROSERVICES

ADAM SANDOR - CONTAINER SOLUTIONS

AGE OF NETFLIX ENGINEERING

Independent teams

No code reuse between components

Independent databases

Independent release & deploy

But what if I
need to scale to a
million...

You're not
Netflix!





SCALE

VELOCITY

RESILIENCE

COST OPTIMIZATION

SMALL SCALE

Low traffic

Single or few teams

Downtime is OK

Costs are not a concern

DEMONS OF MICROSERVICES



MONOLITH



MACHINES



SELF HEALING



ORGANISATIONAL SILOS



CLOUD NATIVE WEAPONRY



DOCKER



Slices VMs into tiny pieces

DOCKER-COMPOSE



Easy to use realistic development

DC/OS



Orchestrate all the things



Prometheus

Stackdriver

Sysdig

CoScale

DataDog



GitLab

Jenkins

CodeShip

OpenShift





E1M7: Monitoring

E1M6: DevOps

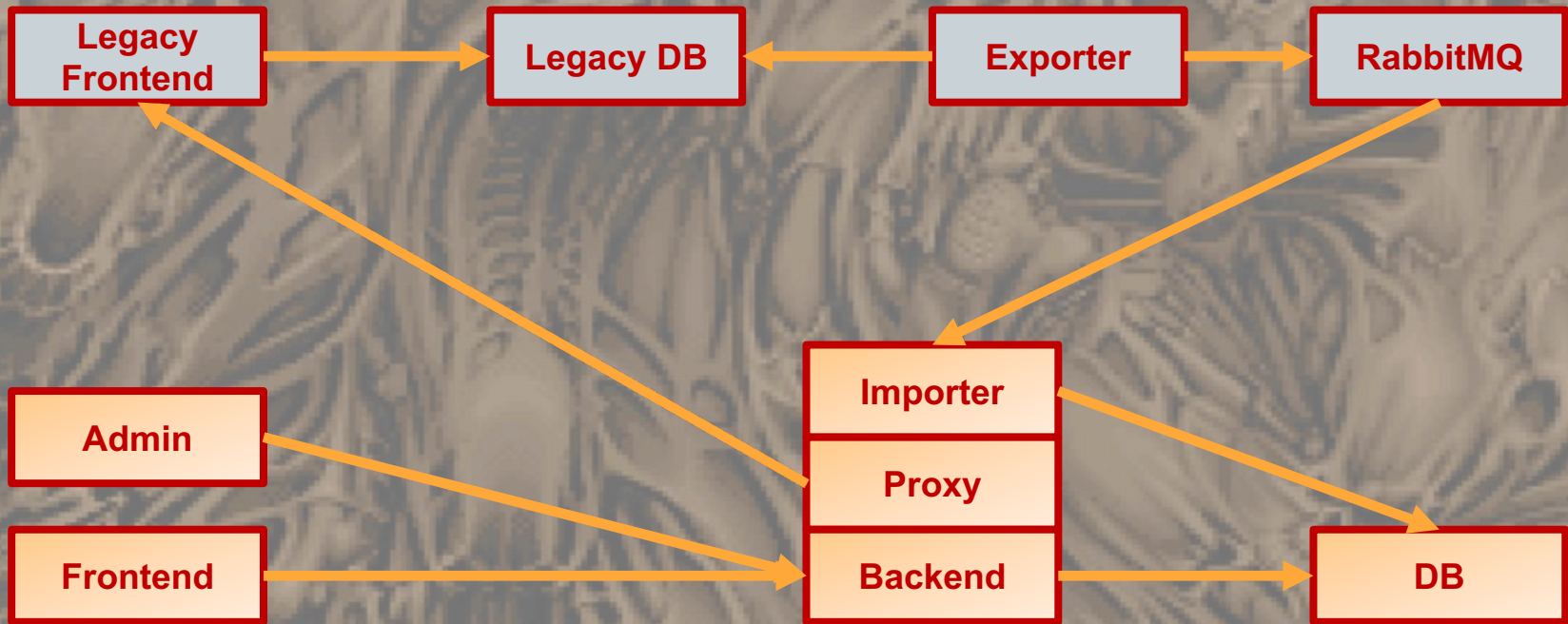
E1M5: Training

E1M4: Production
Workload

E1M3: CI/CD

E1M2: Managing
Environments

E1M1:
Independent
Delivery



INDEPENDENT DELIVERY

Separate source

Separate

Separate not

Own your database!



**Deployment
Service
HPA**

Admin

Admin

**Deployment
Service
HPA**

Frontend

Frontend

**Deployment
Service
HPA**

Backend

Backend

Backend

**Managed
Service**

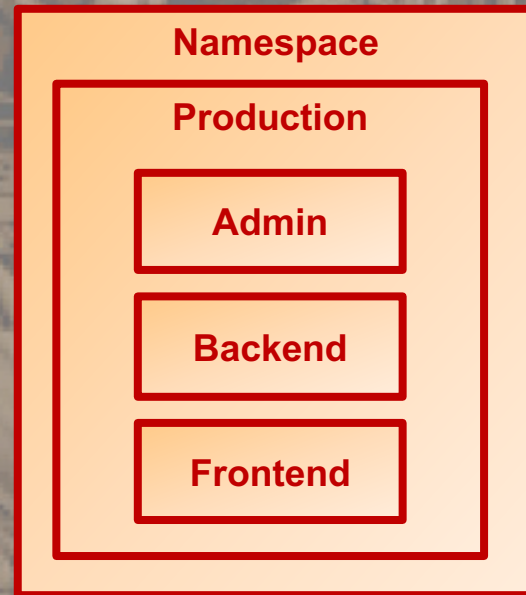
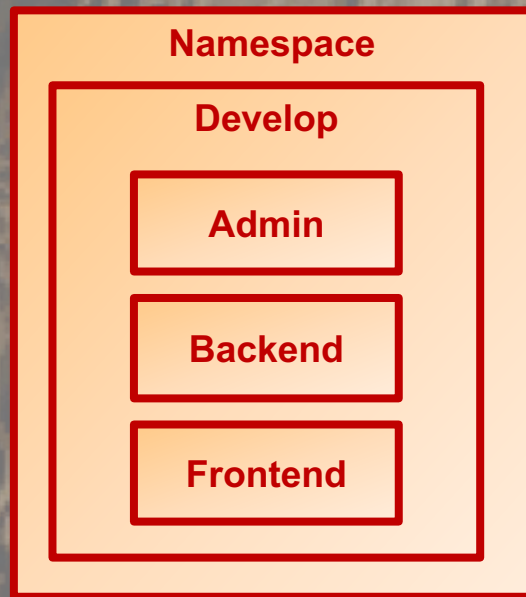
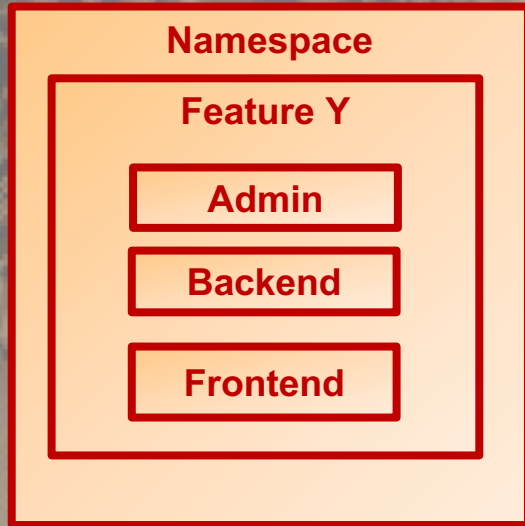
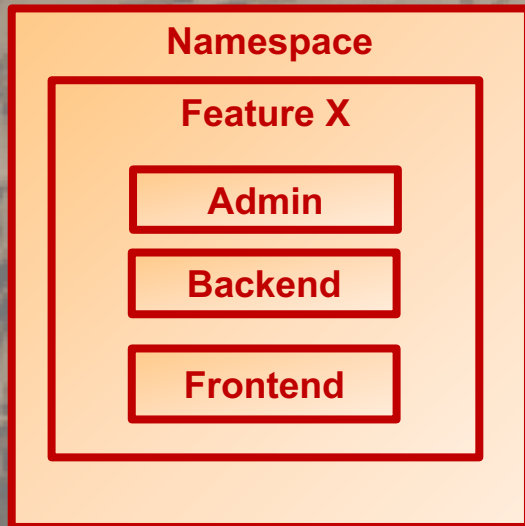
DB

MANAGING ENVIRONMENTS

Cheap

Disposable

Versioned



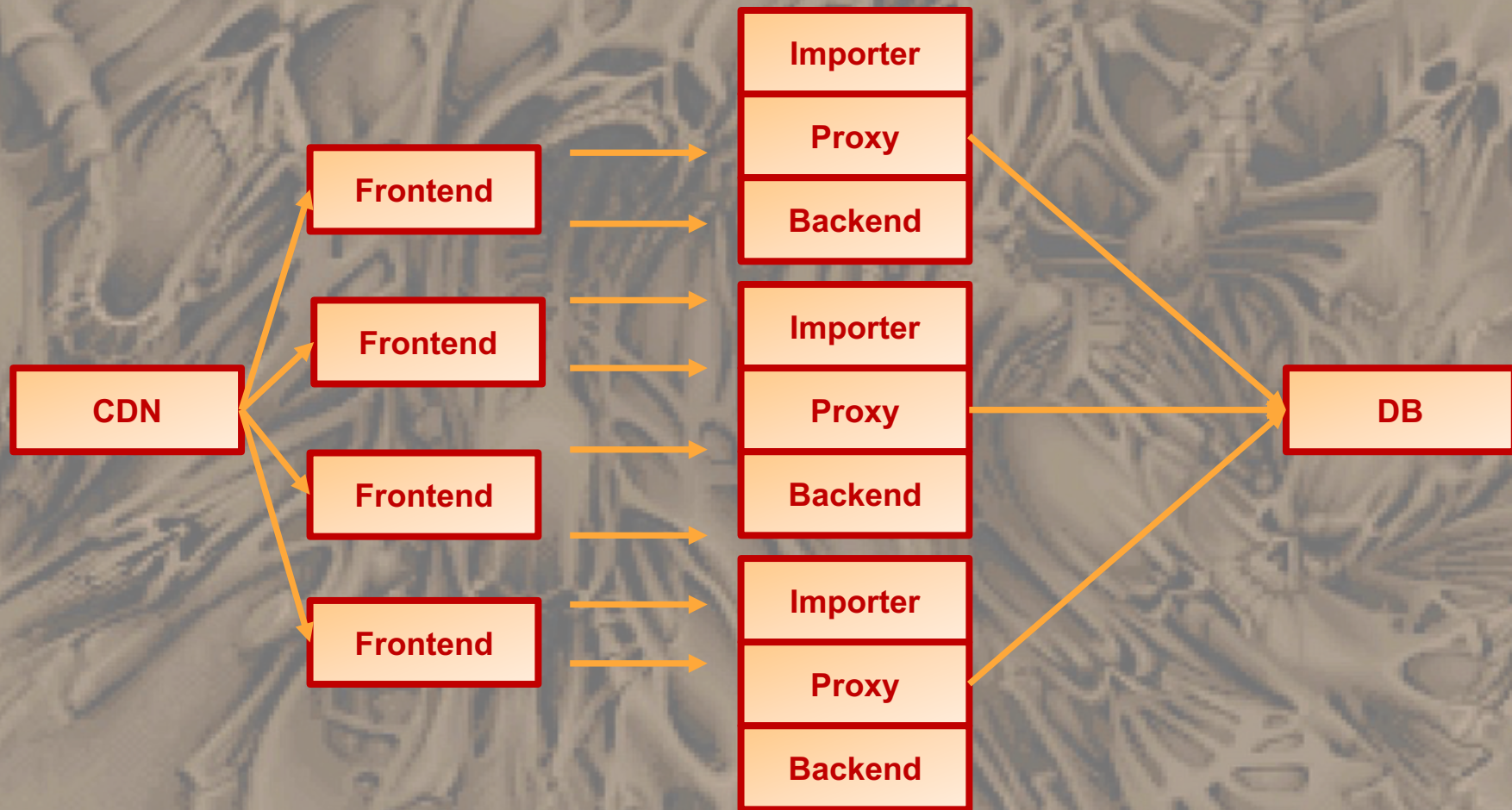
PRODUCTION WORKLOAD



Autoscaling

Resource limit / reservations

Finetuning



DEVOPS

OPS is a necessity

Develop tooling

Deployments are DEV responsibility

Environments are DEV responsibility

TRAINING

OPS deep platform knowledge

DEV confident as users of the platform

CONTINUOUS DELIVERY



Docker support in build tool

How to push artifacts to environments?

OPS team only creates tooling

MONITORING

OPS only delivers tooling

Inter-service debugging

Log enough but not too much



