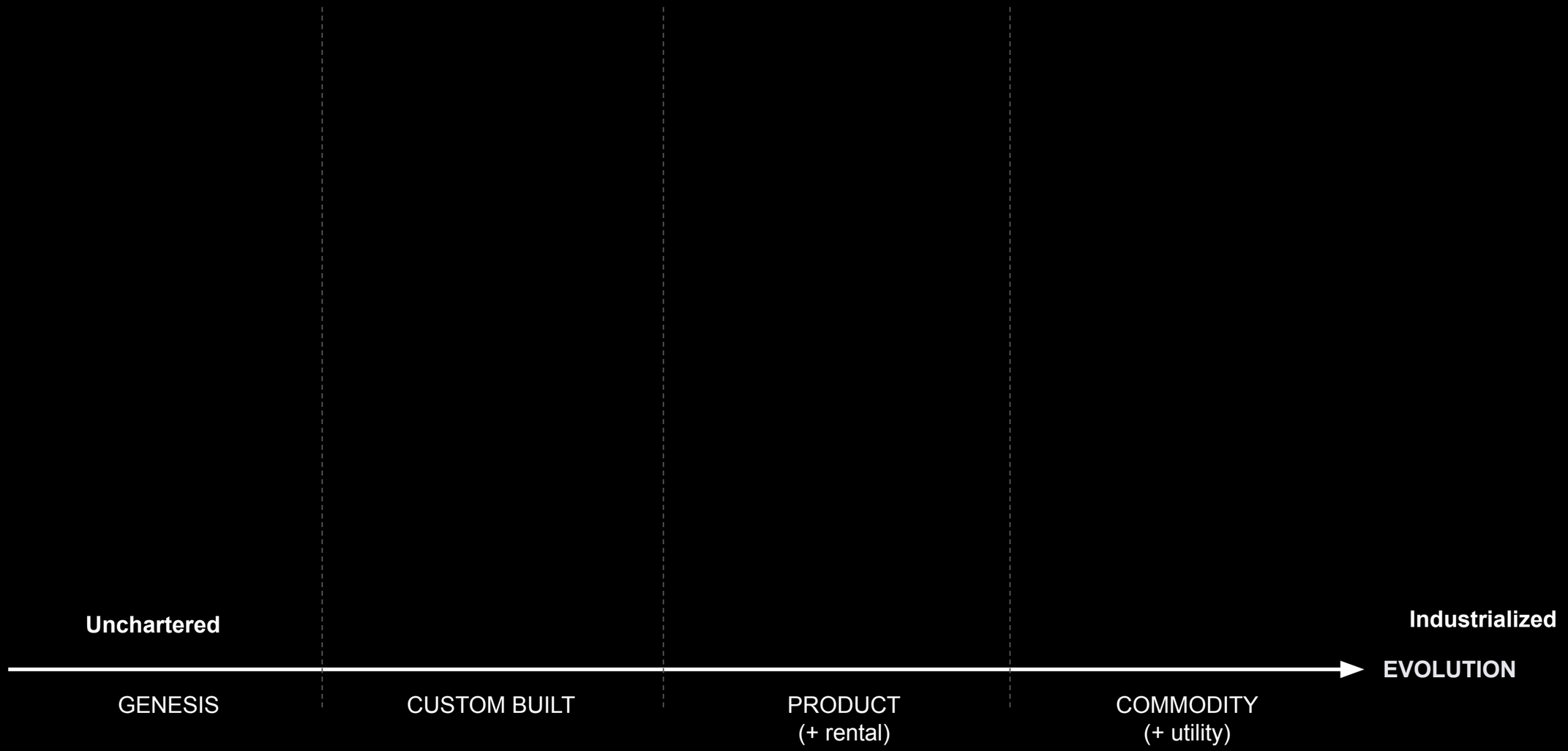
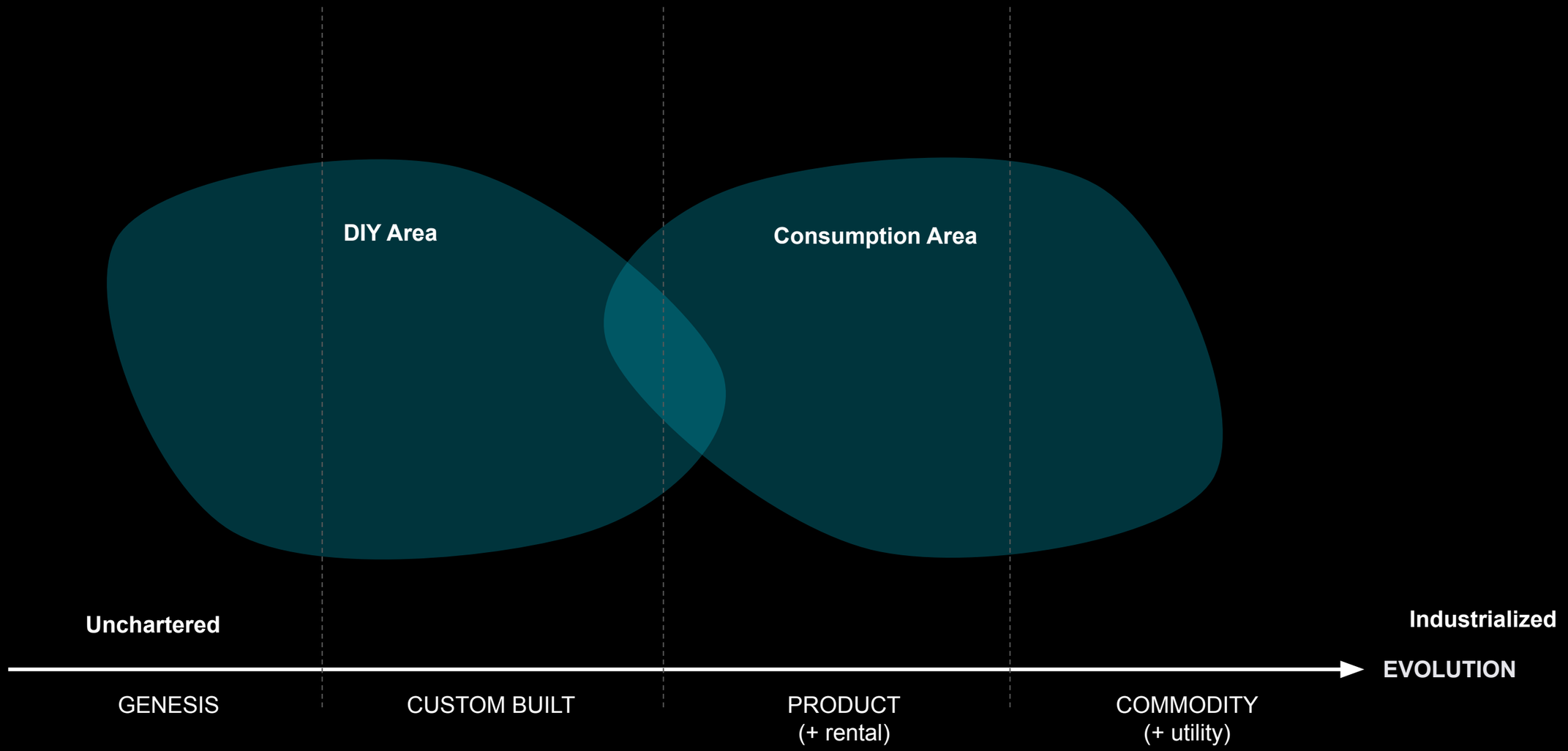
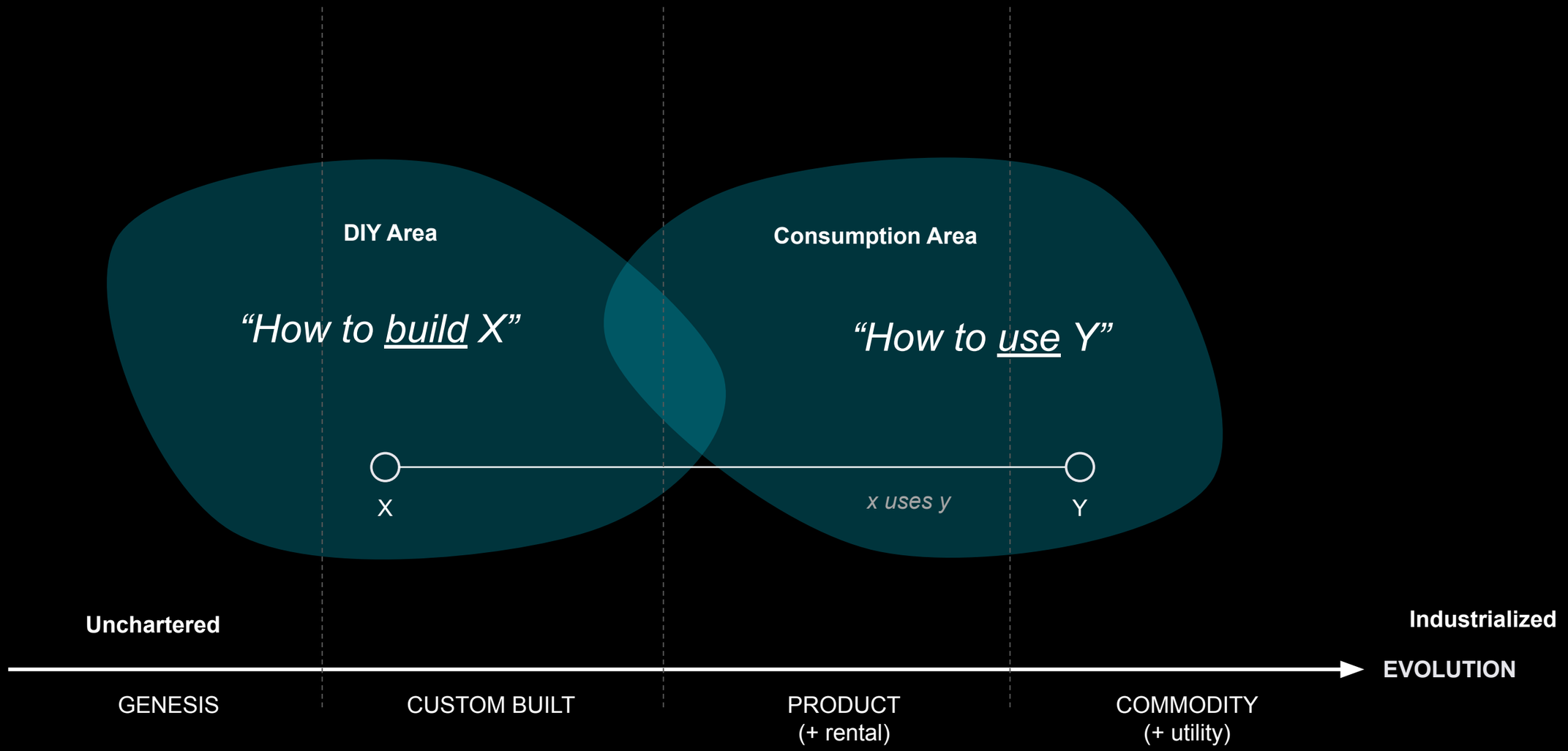
“Do it Yourself” for software engineers

How software engineers can experiment and learn by making architectural decisions

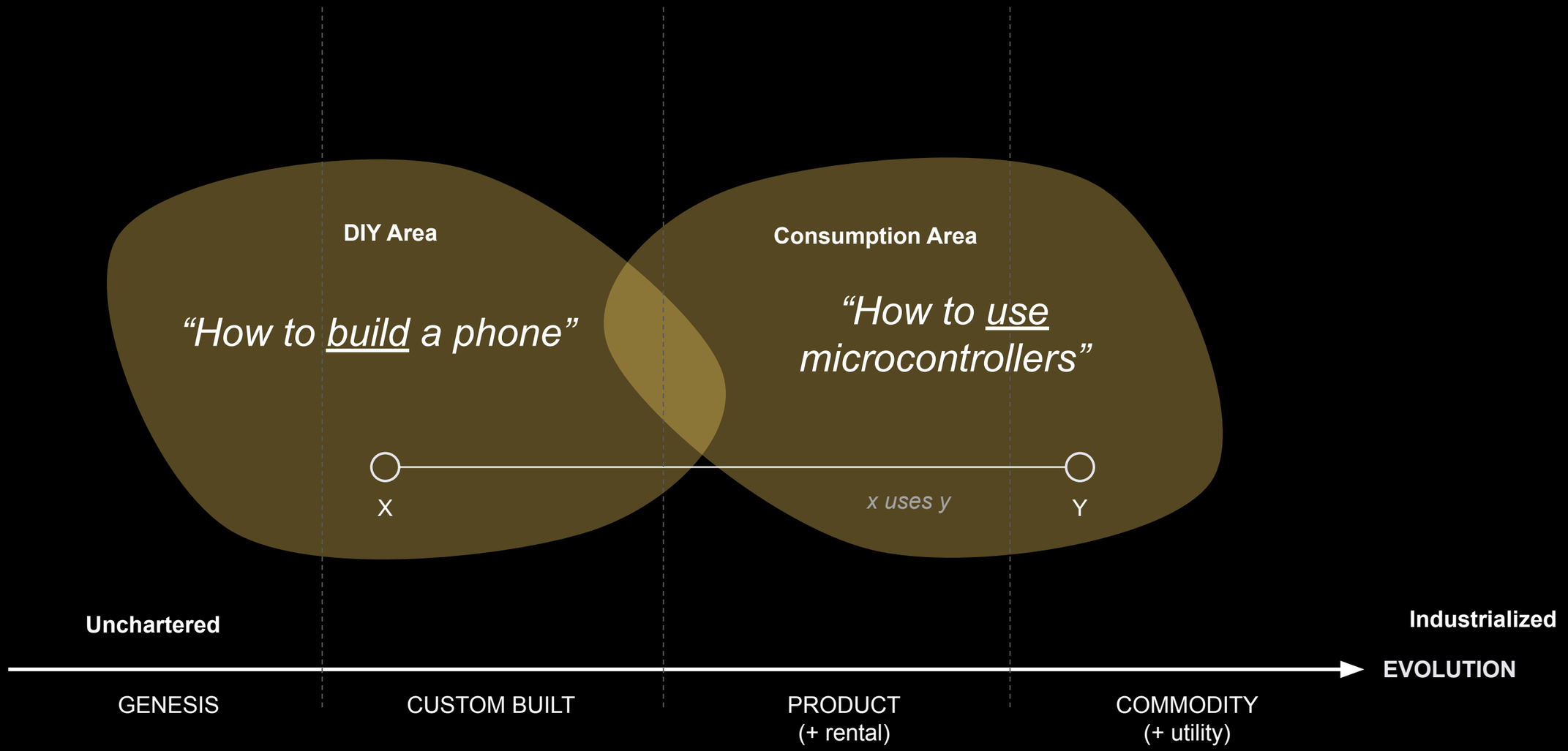
Picture by Anders Jildén via Unsplash


Uncharted
Industrialized
EVOLUTION
GENESIS
CUSTOM BUILT
PRODUCT
(+ rental)
COMMODITY
(+ utility)

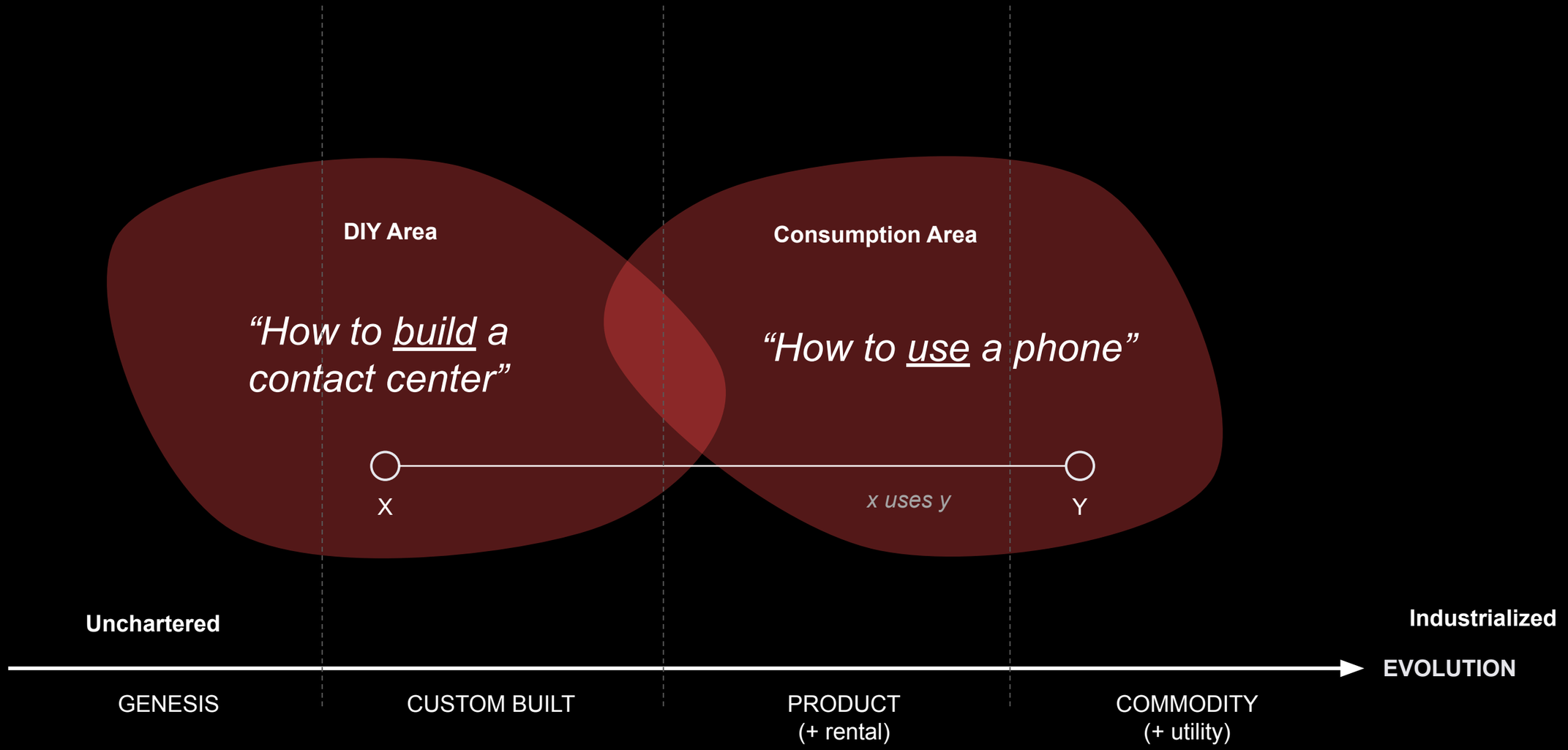

DIY Area
Consumption Area
Unchartered
Industrialized
EVOLUTION
GENESIS
CUSTOM BUILT
PRODUCT
(+ rental)
COMMODITY
(+ utility)


DIY Area
Consumption Area
“How to build X”
“How to use Y”
X
x uses y
Y
Unchartered
Industrialized
EVOLUTION
GENESIS
CUSTOM BUILT
PRODUCT
(+ rental)
COMMODITY
(+ utility)

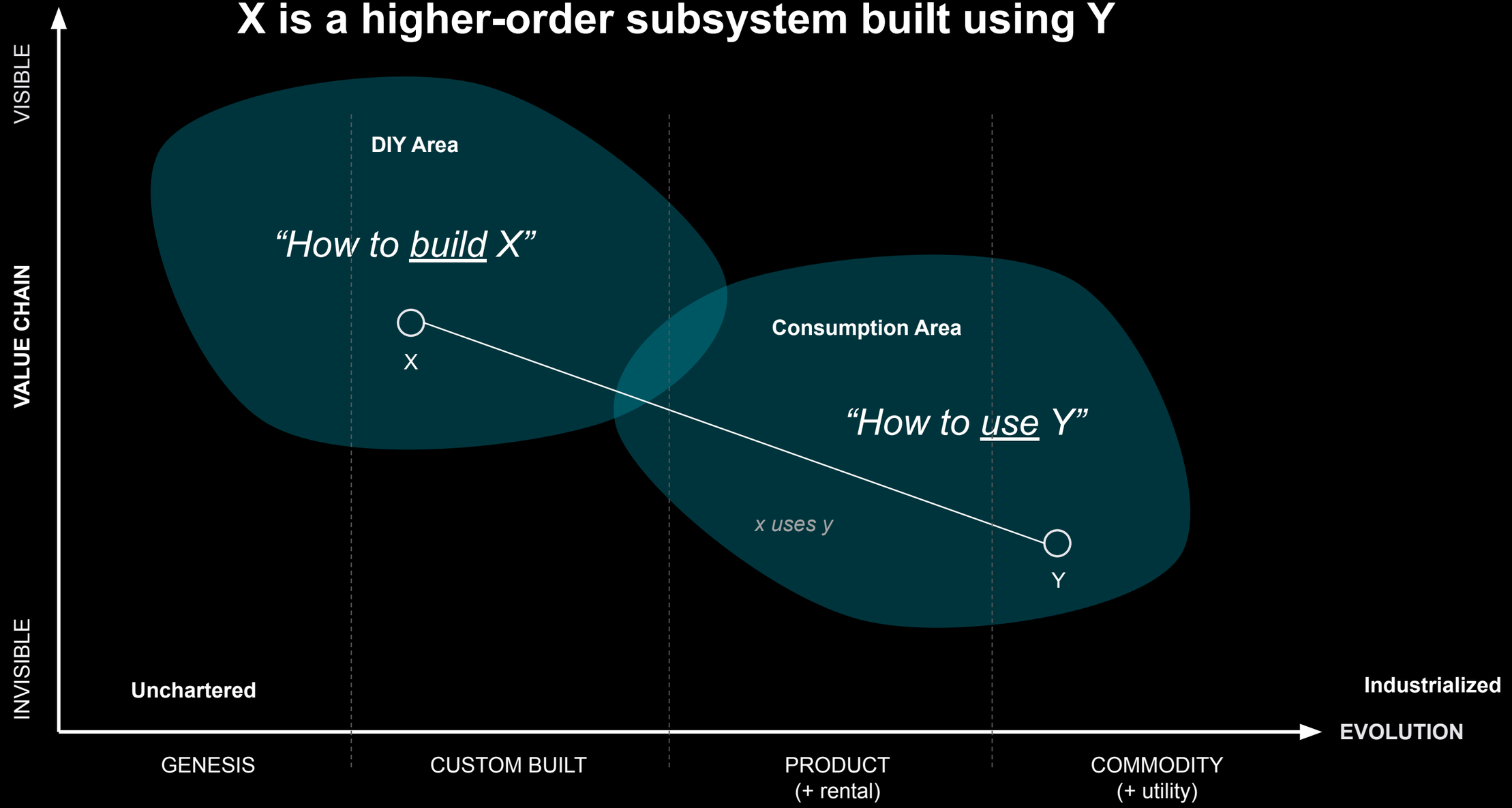
How it started …



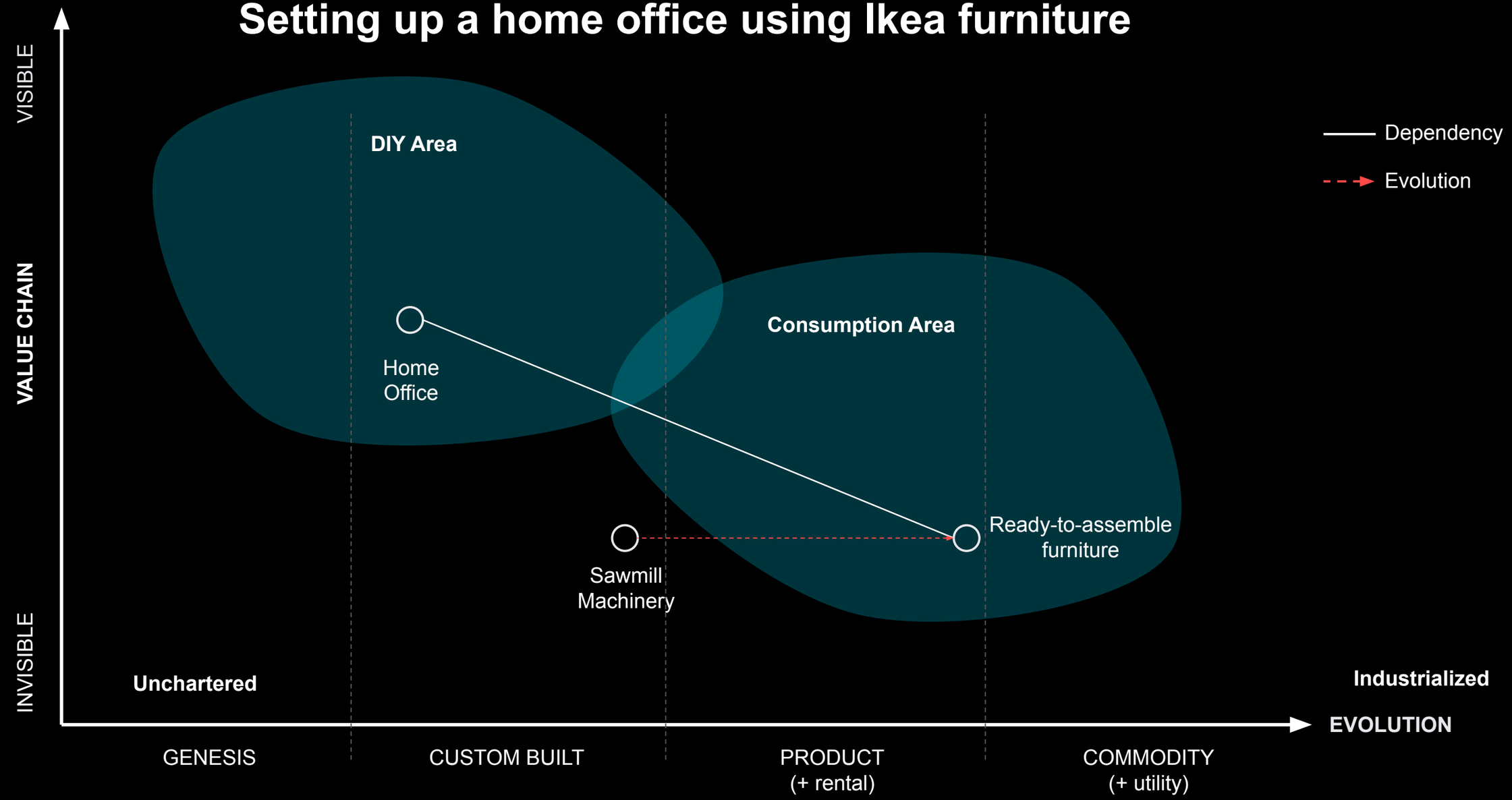
How it ended …


DIY Area
"How to build a contact center"
Consumption Area
"How to use a phone"
X
x uses y
Y
Unchartered
Industrialized
EVOLUTION
GENESIS
CUSTOM BUILT
PRODUCT
(+ rental)
COMMODITY
(+ utility)

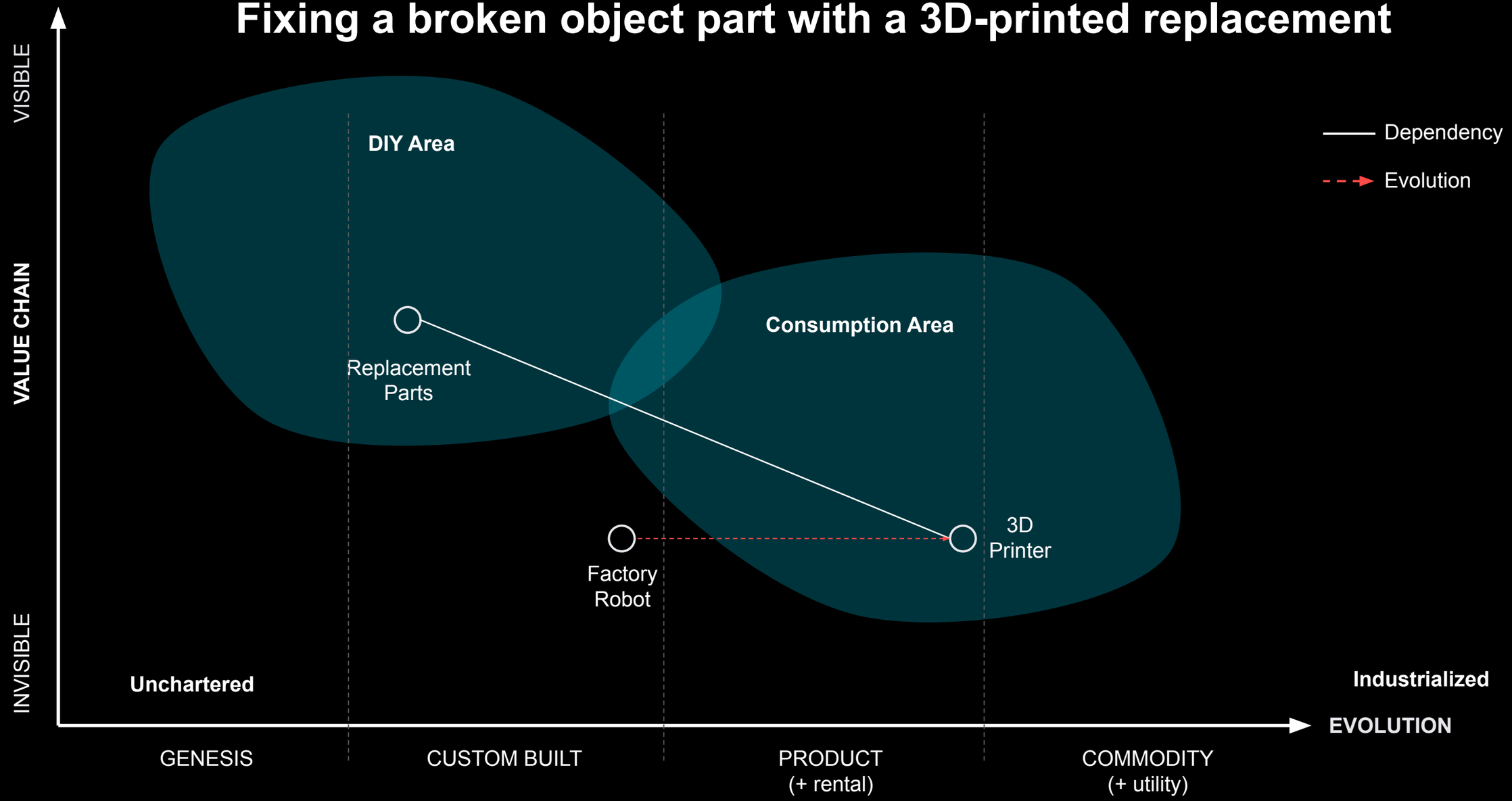
X is a higher-order subsystem built using Y



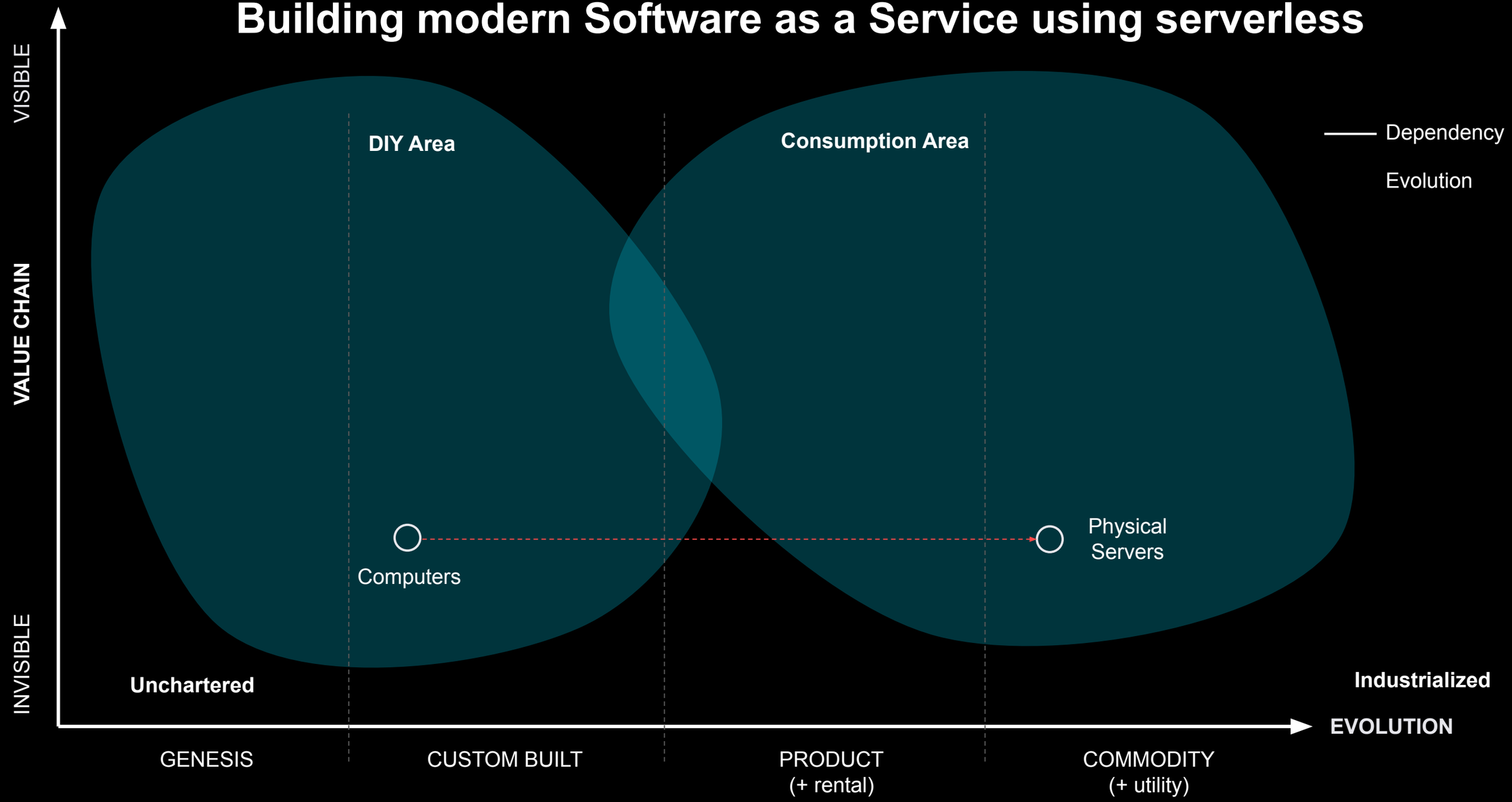
Setting up a home office using Ikea furniture



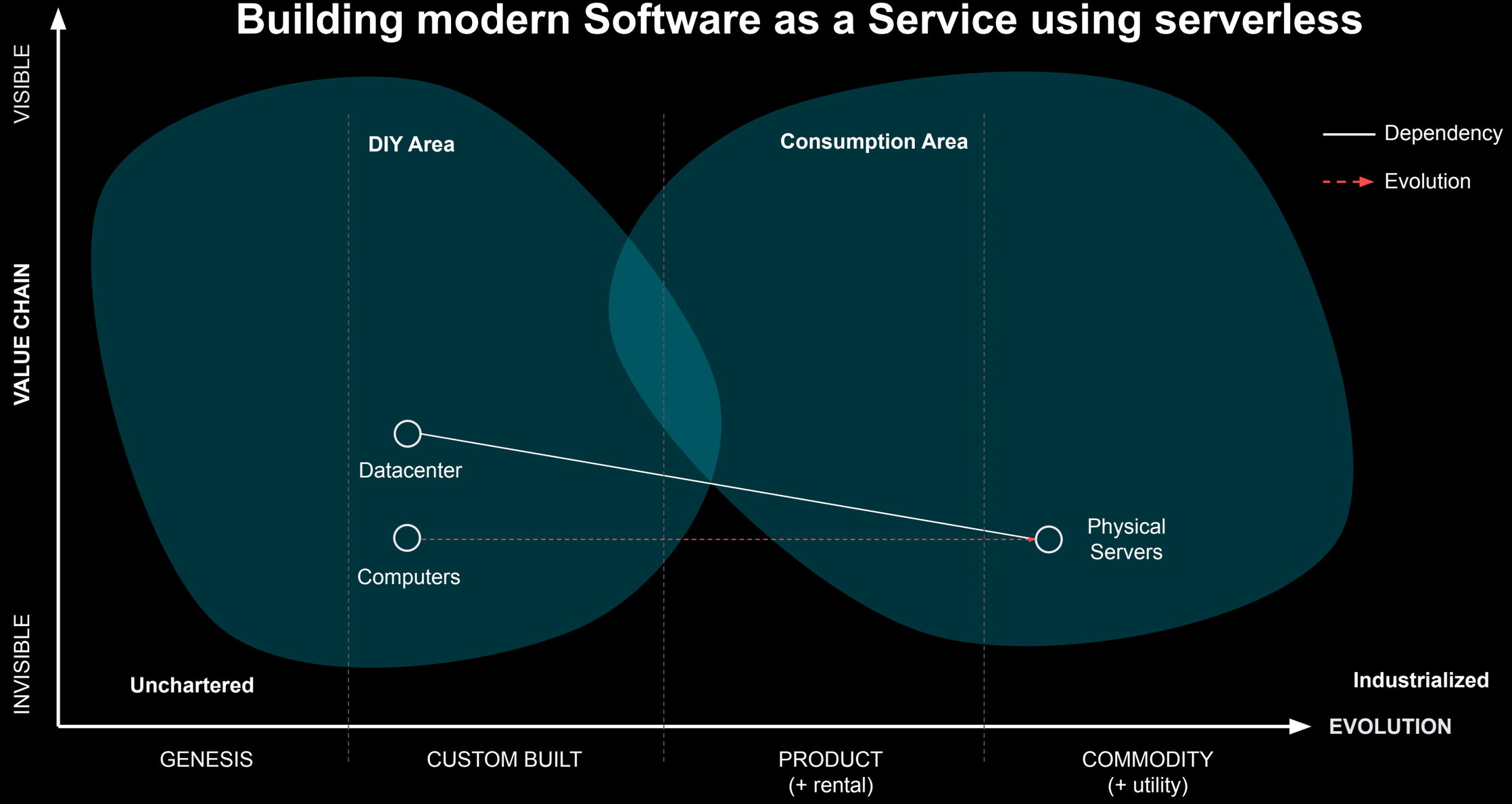
Fixing a broken object part with a 3D-printed replacement



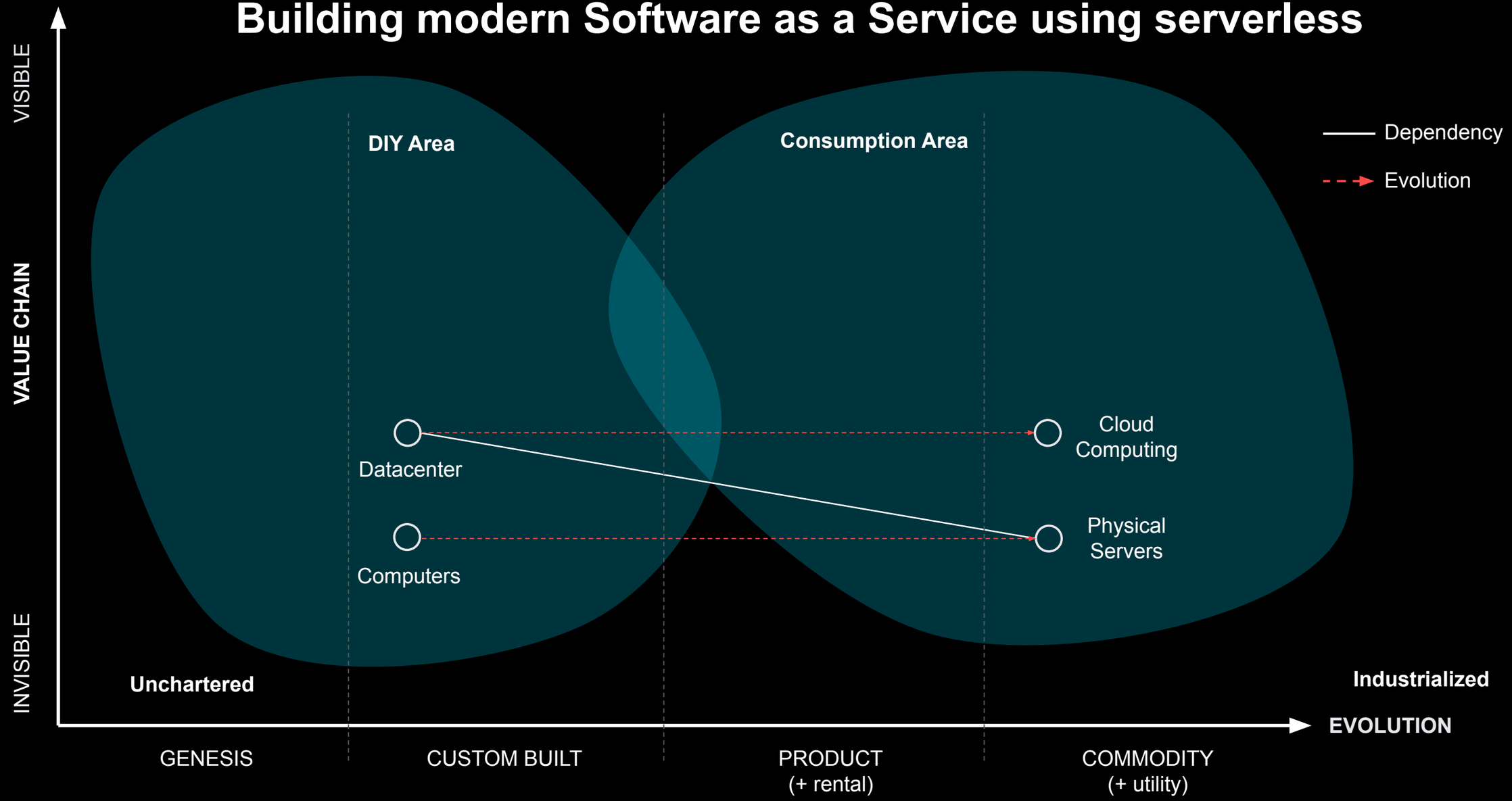
Building modern Software as a Service using serverless



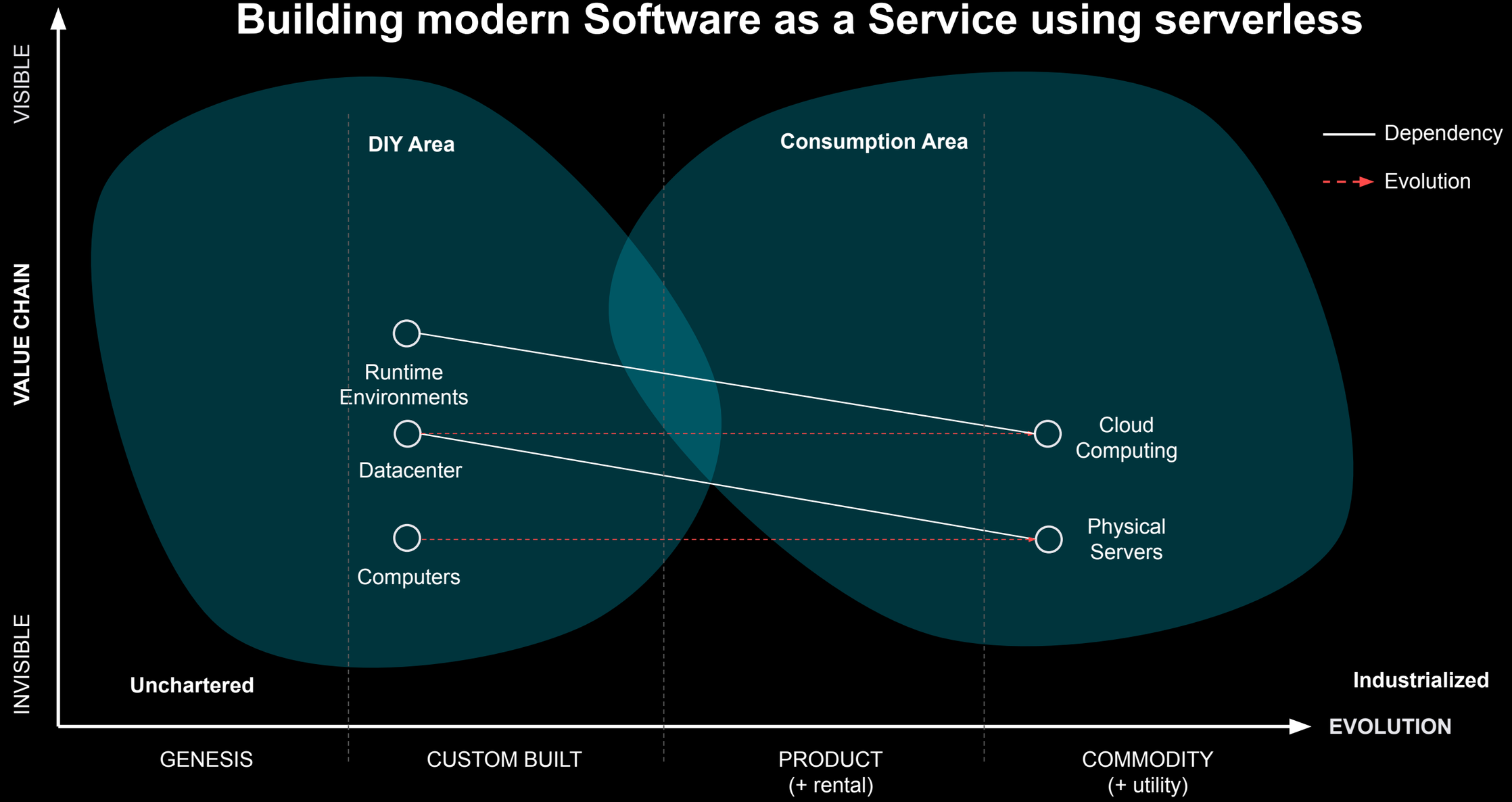
Building modern Software as a Service using serverless



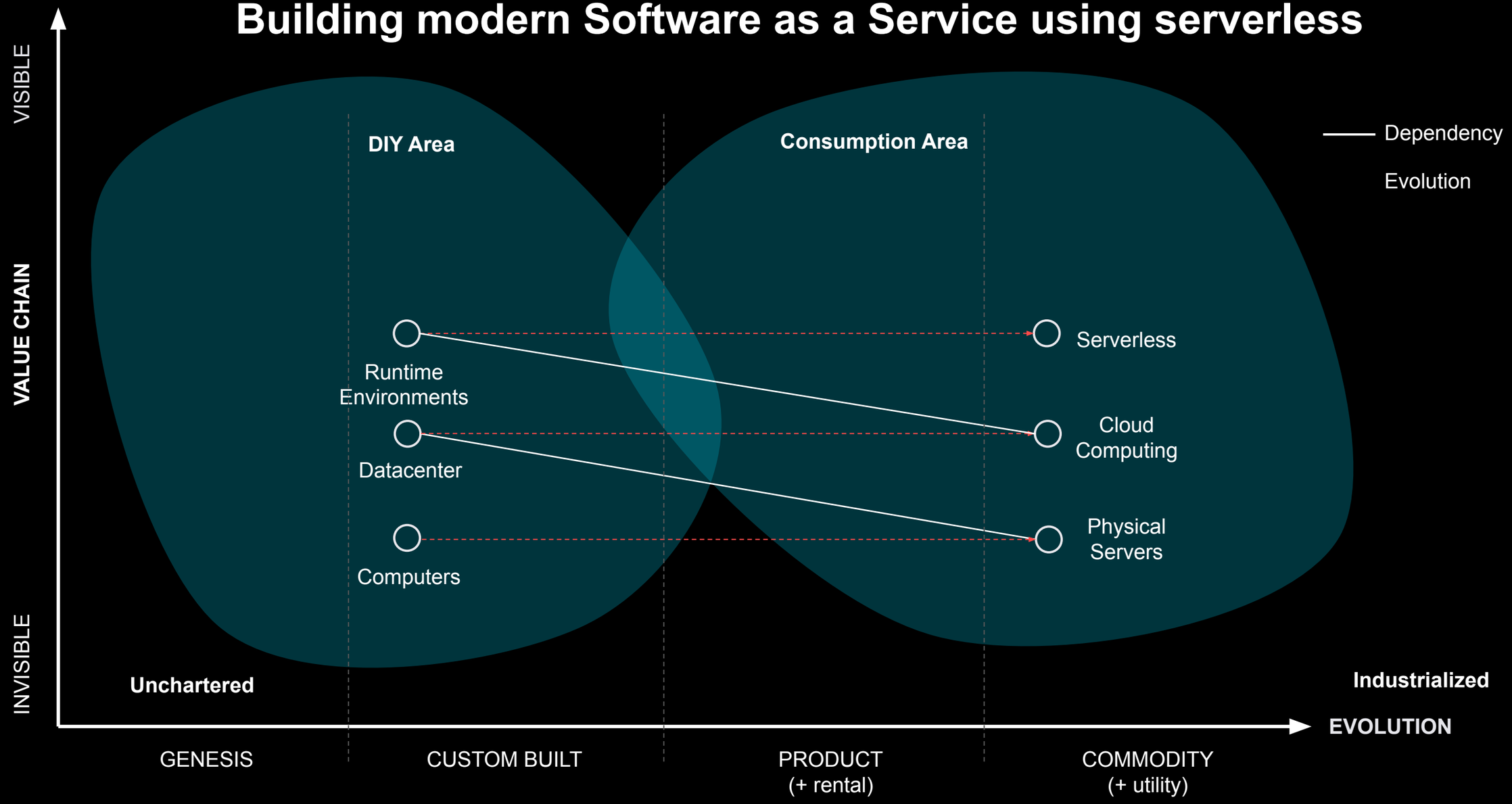
Building modern Software as a Service using serverless



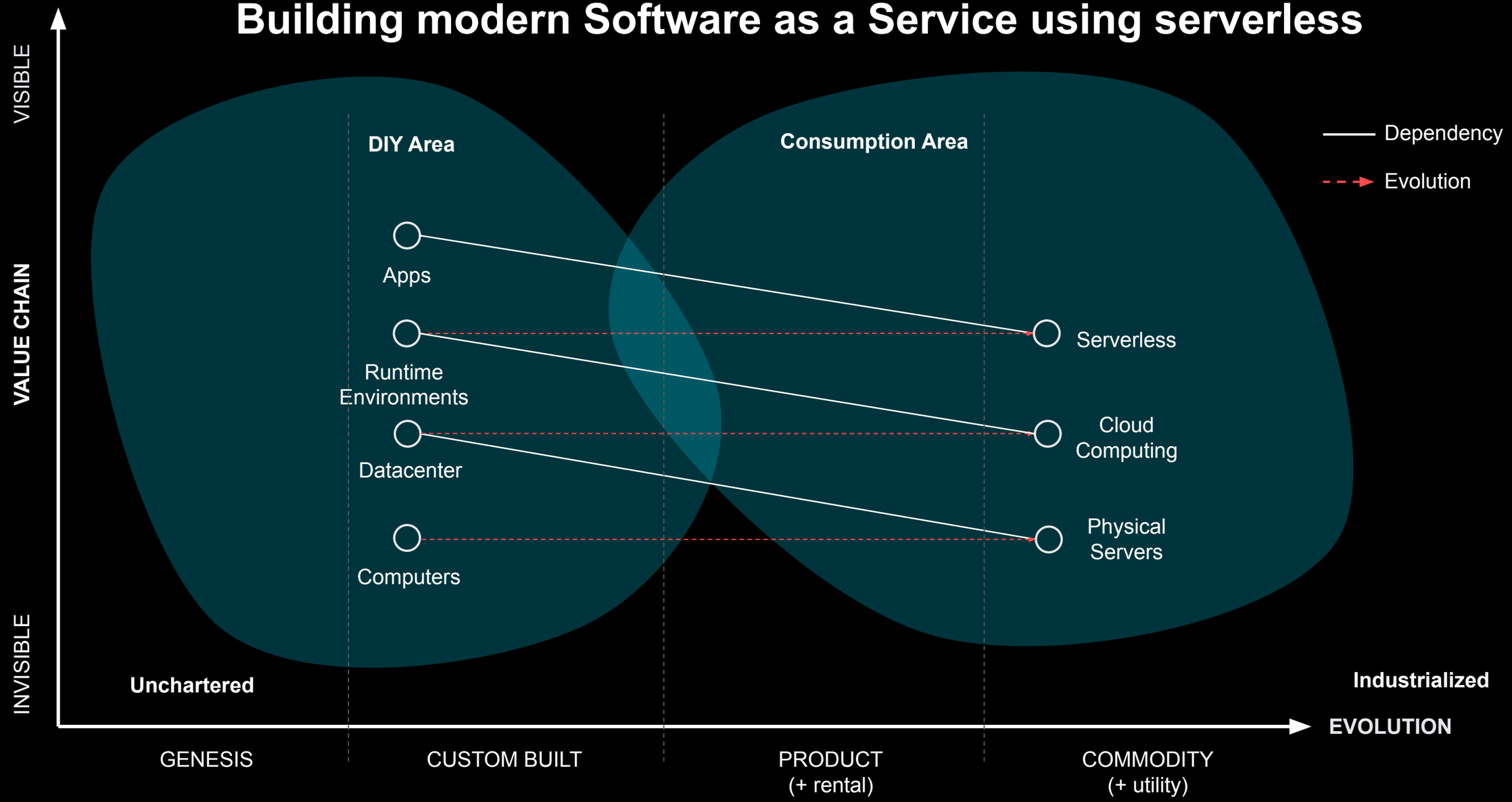
Building modern Software as a Service using serverless

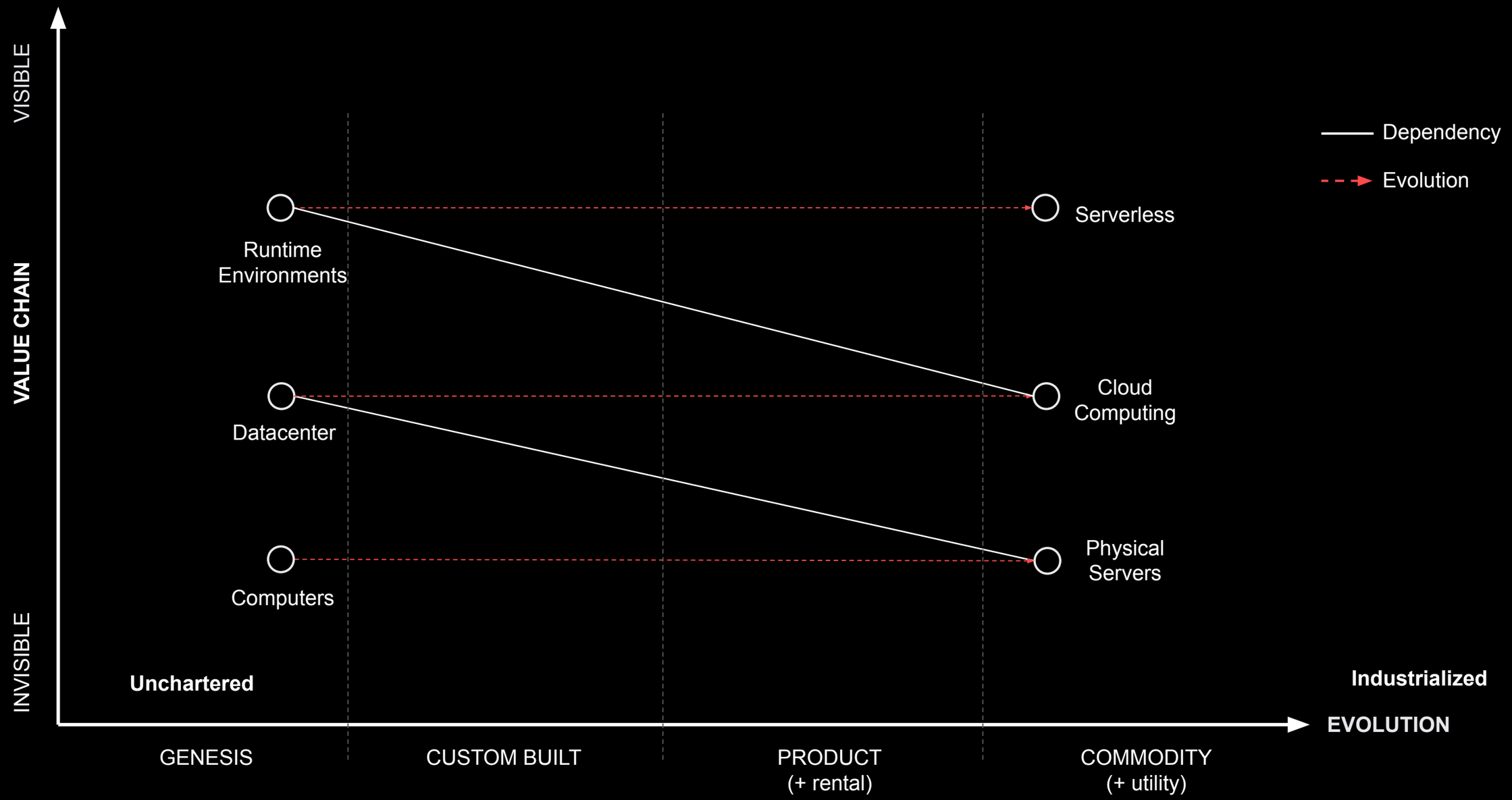


Building modern Software as a Service using serverless



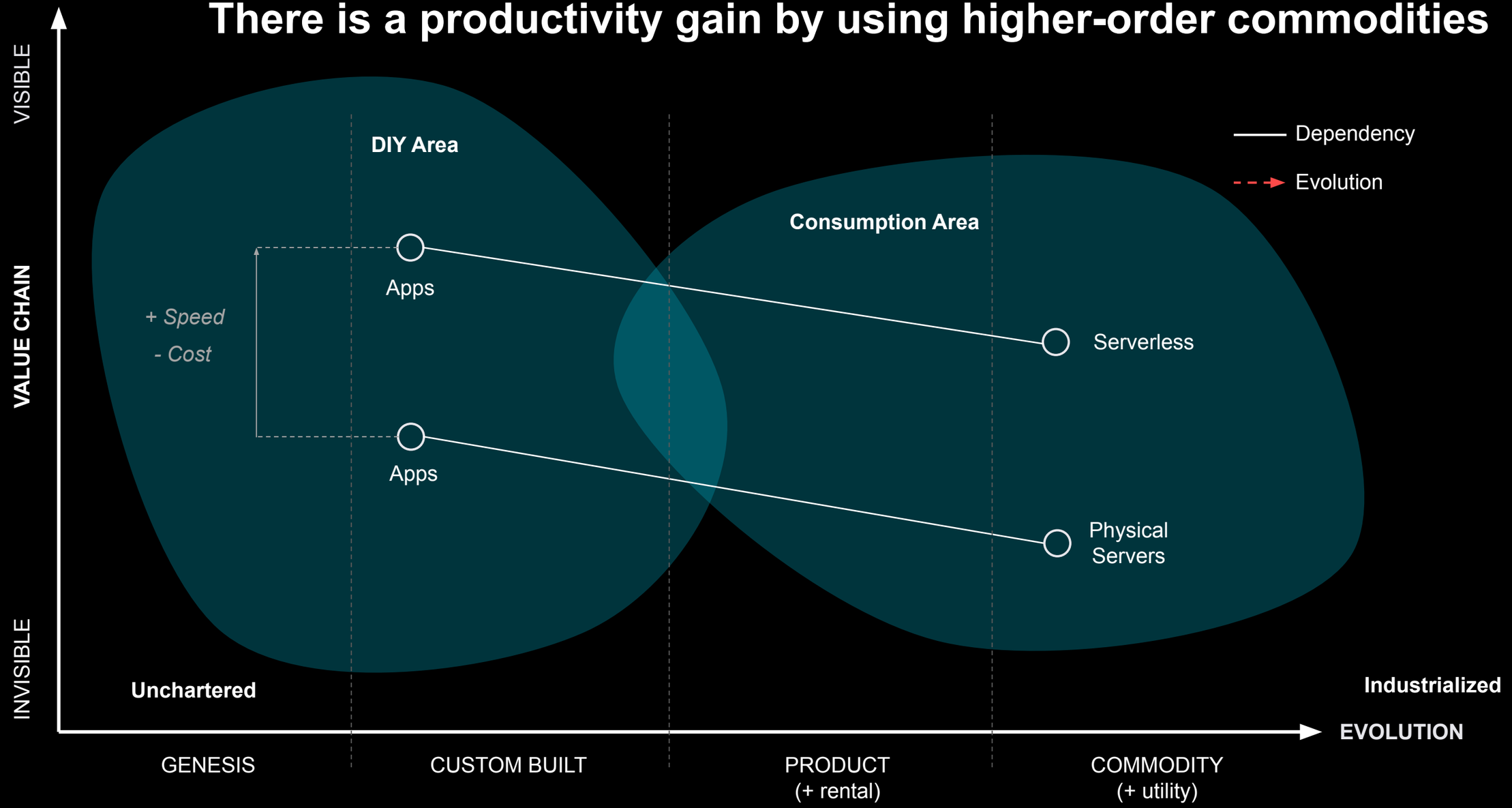
Building modern Software as a Service using serverless




VISIBLE
VALUE CHAIN
INVISIBLE
Runtime Environments
Datacenter
Computers
Serverless
Cloud Computing
Physical Servers
Dependency
Evolution
Unchartered
Industrialized
EVOLUTION
GENESIS
CUSTOM BUILT
PRODUCT (+ rental)
COMMODITY (+ utility)

But we had apps before serverless!

There is a productivity gain by using higher-order commodities

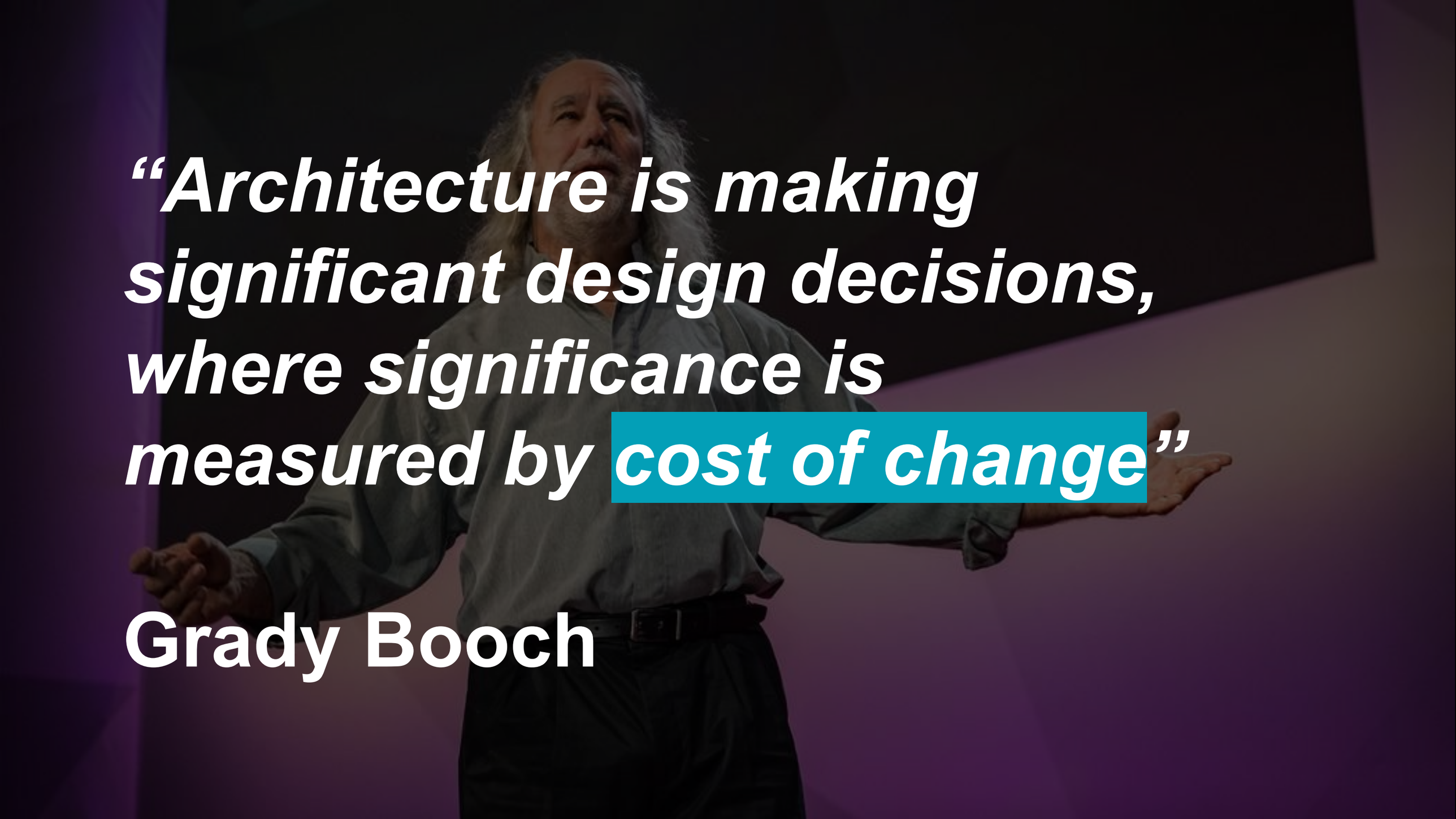


You also had home offices before Ikea and replacements before 3D printers.

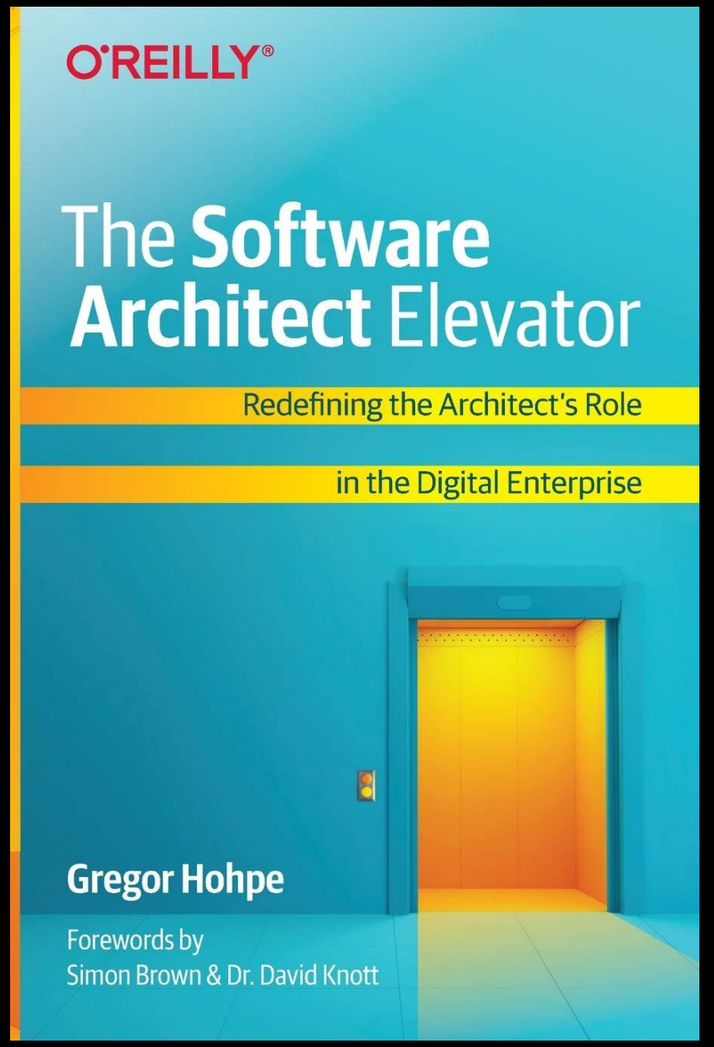
You just had them at **less speed and more cost.**

Picture by Semyon Borisov via Unsplash

More speed and less cost?


“Architecture is making significant design decisions, where significance is measured by cost of change”
Grady Booch

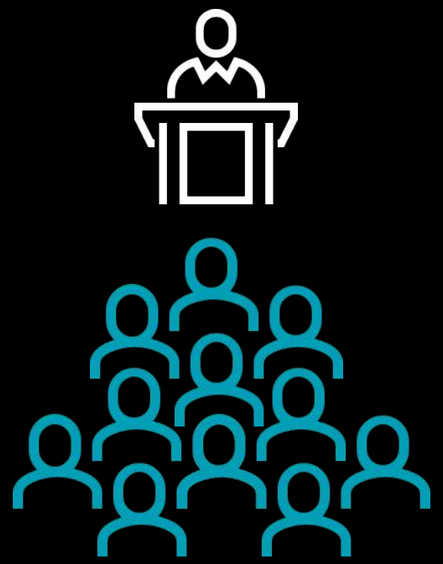
The Architect Elevator – Do you need architects with your architecture?



Do you need architects with your architecture?

The Architect Elevator – Do you need architects with your architecture?

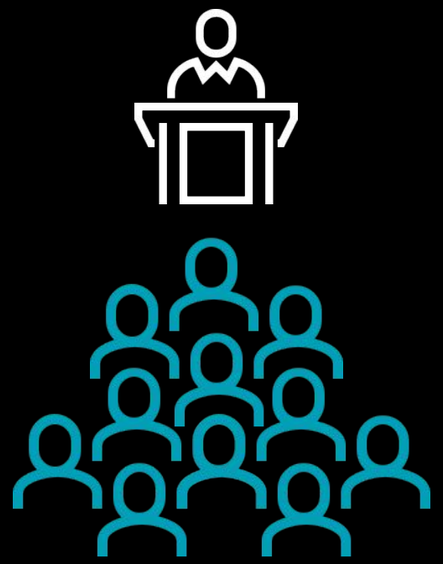
Benevolent Dictator



?

The Architect Elevator – Do you need architects with your architecture?

Benevolent Dictator

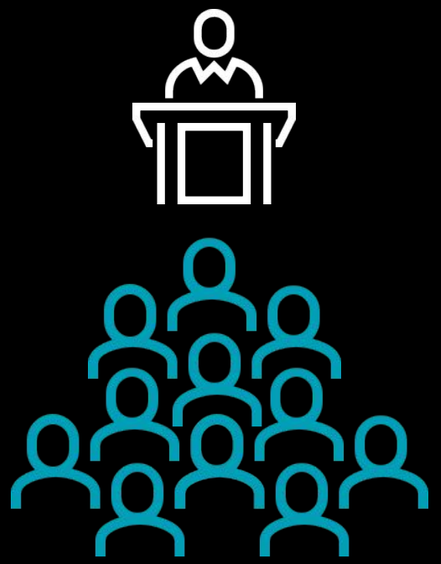


?

Primus inter pares

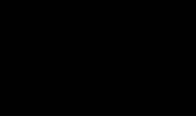
The Architect Elevator – Do you need architects with your architecture?

Benevolent Dictator



?

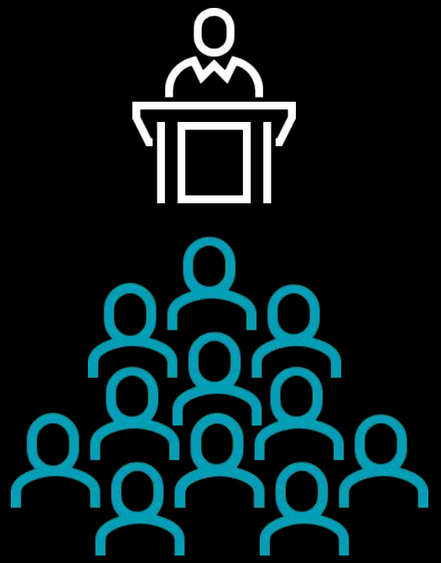
Primus inter pares



Arch without the archs

The Architect Elevator – Do you need architects with your architecture?

Benevolent Dictator



?

Primus inter pares

♥

Arch without the archs

♥

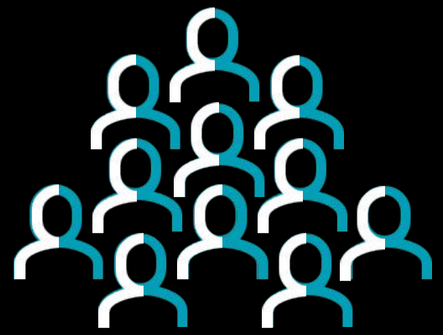
Inmates running asylum

X

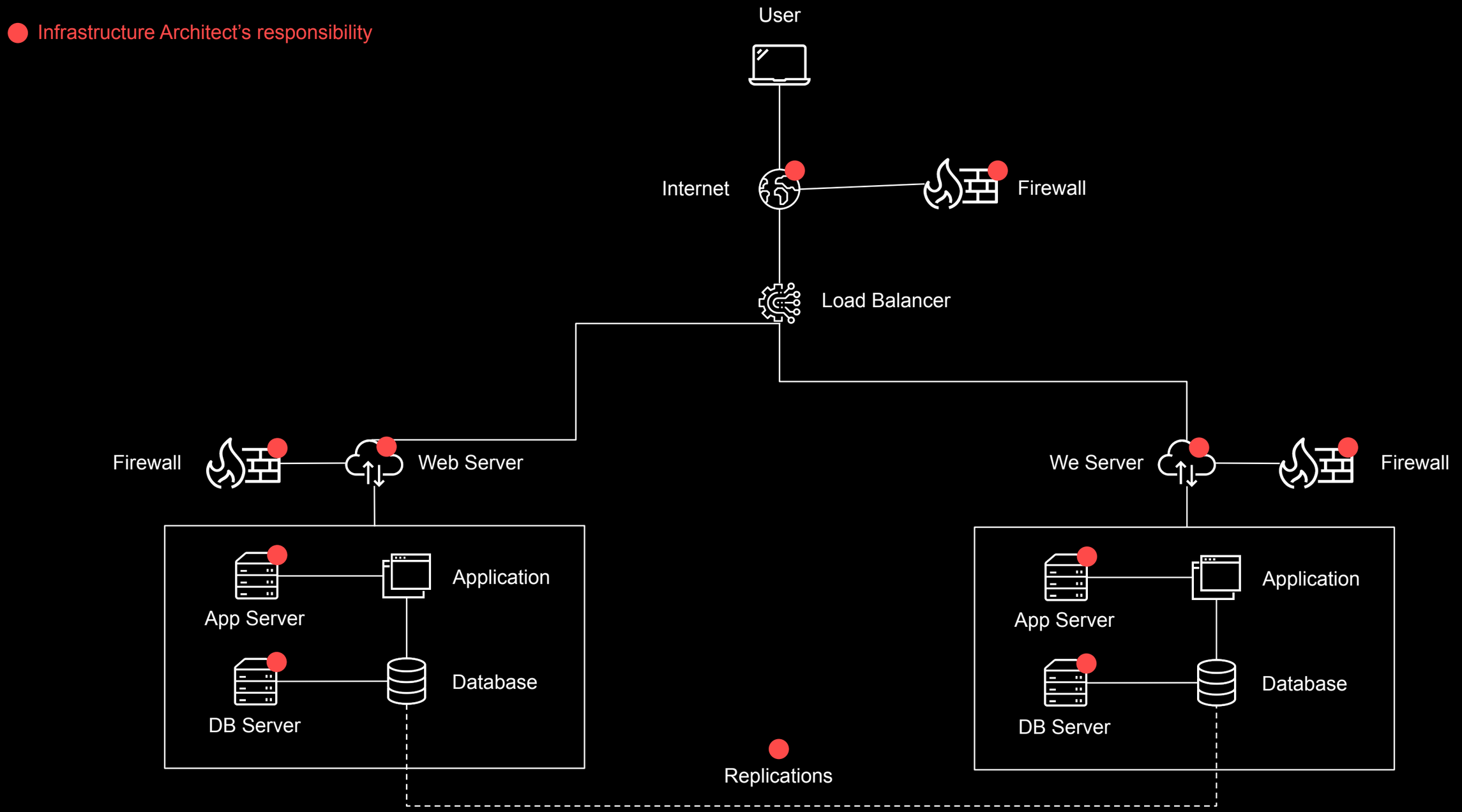
Architecture

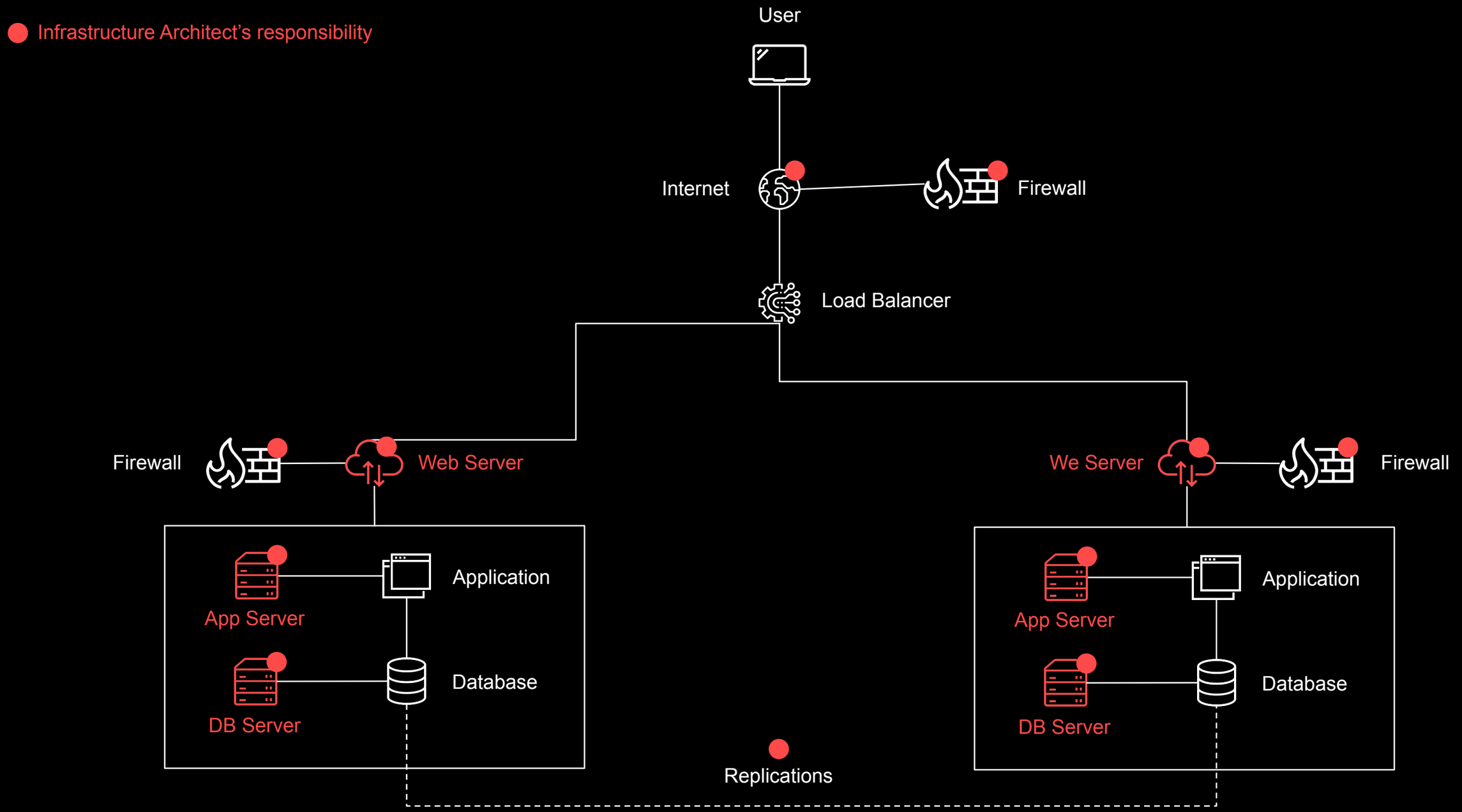
DIY for software engineers

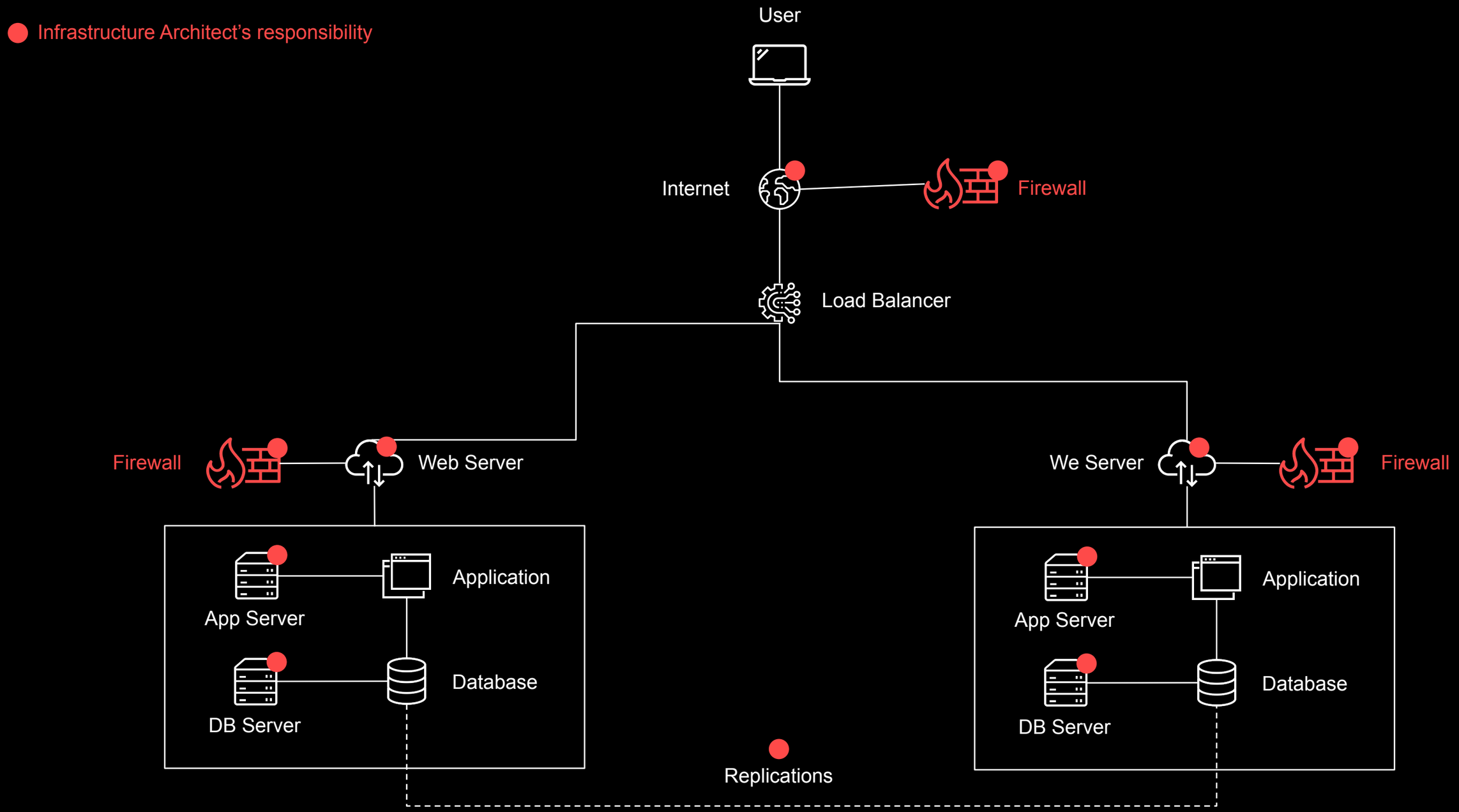
Arch without the archs

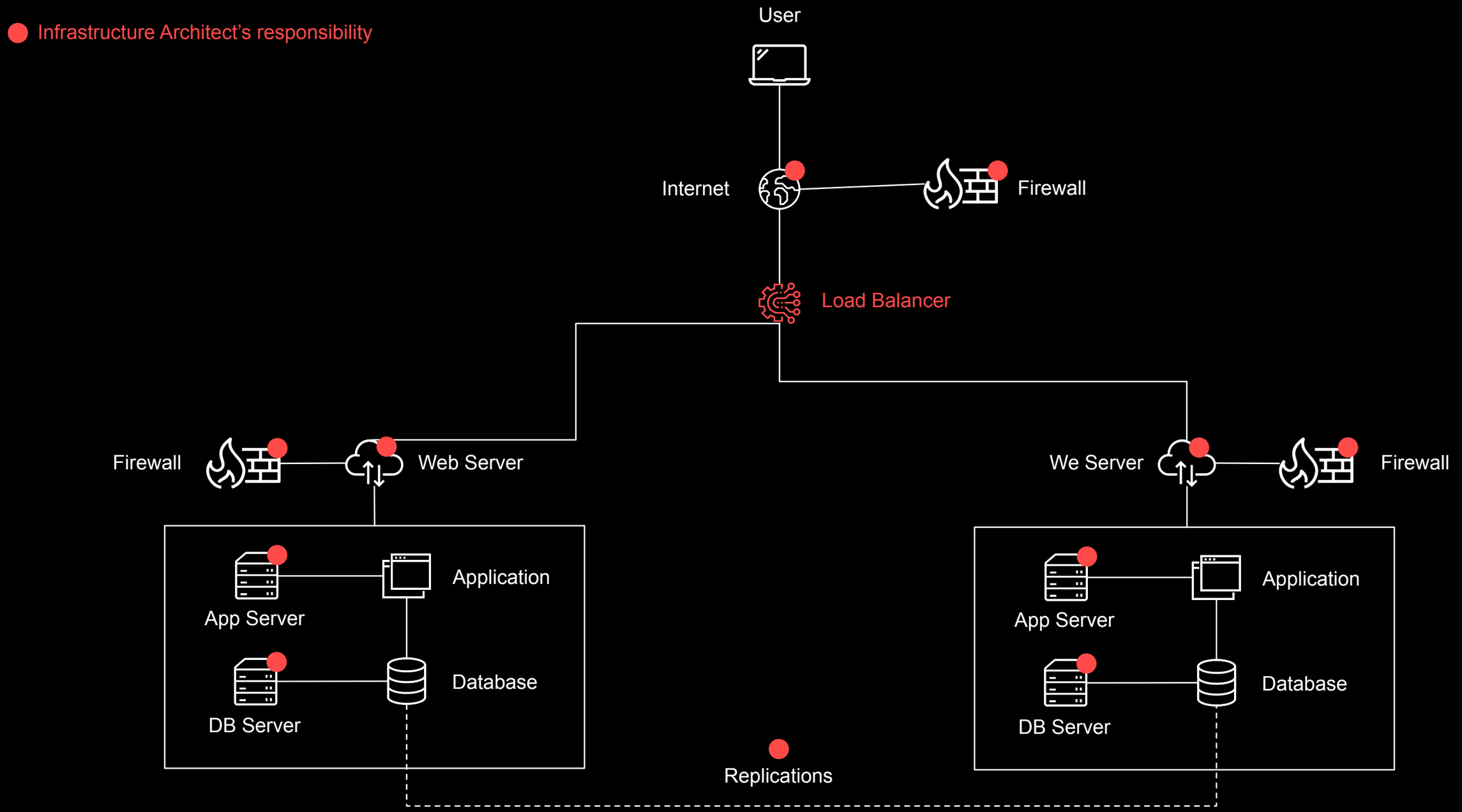


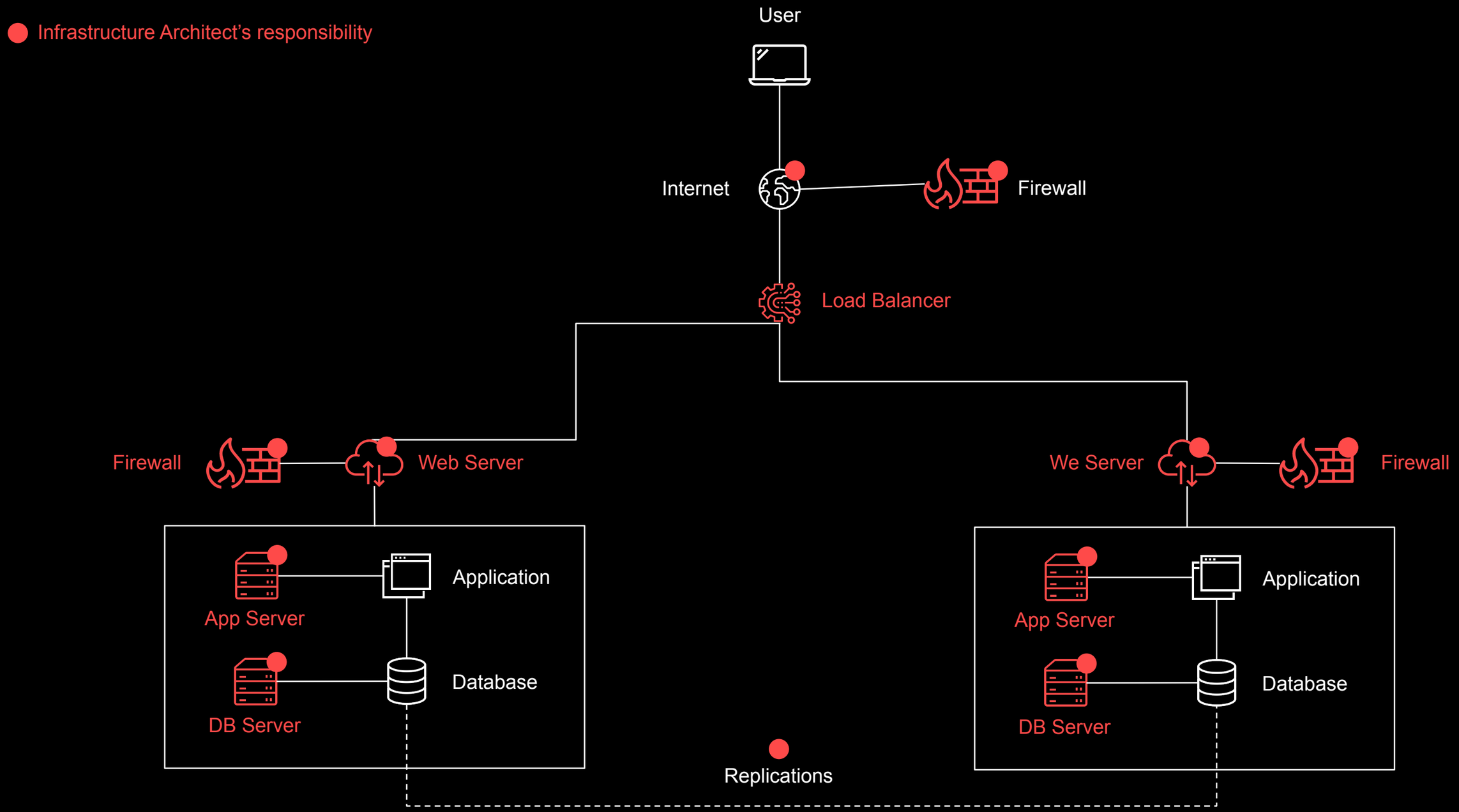
Example with a very simple Web application architecture

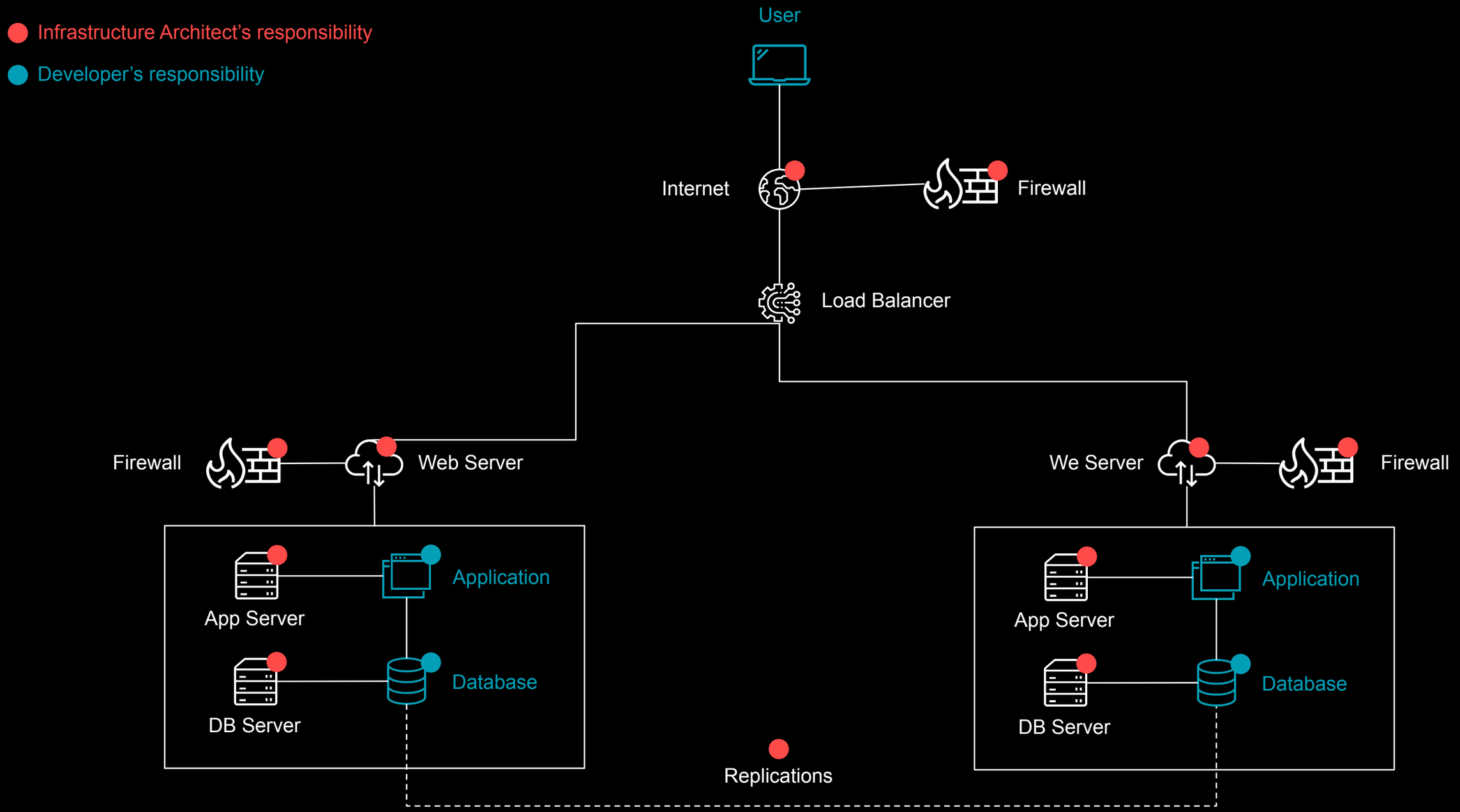

Infrastructure Architect's responsibility
User
Internet
Firewall
Load Balancer
Firewall
Web Server
We Server
Firewall
Application
App Server
Database
DB Server
Application
App Server
Database
DB Server
Replications

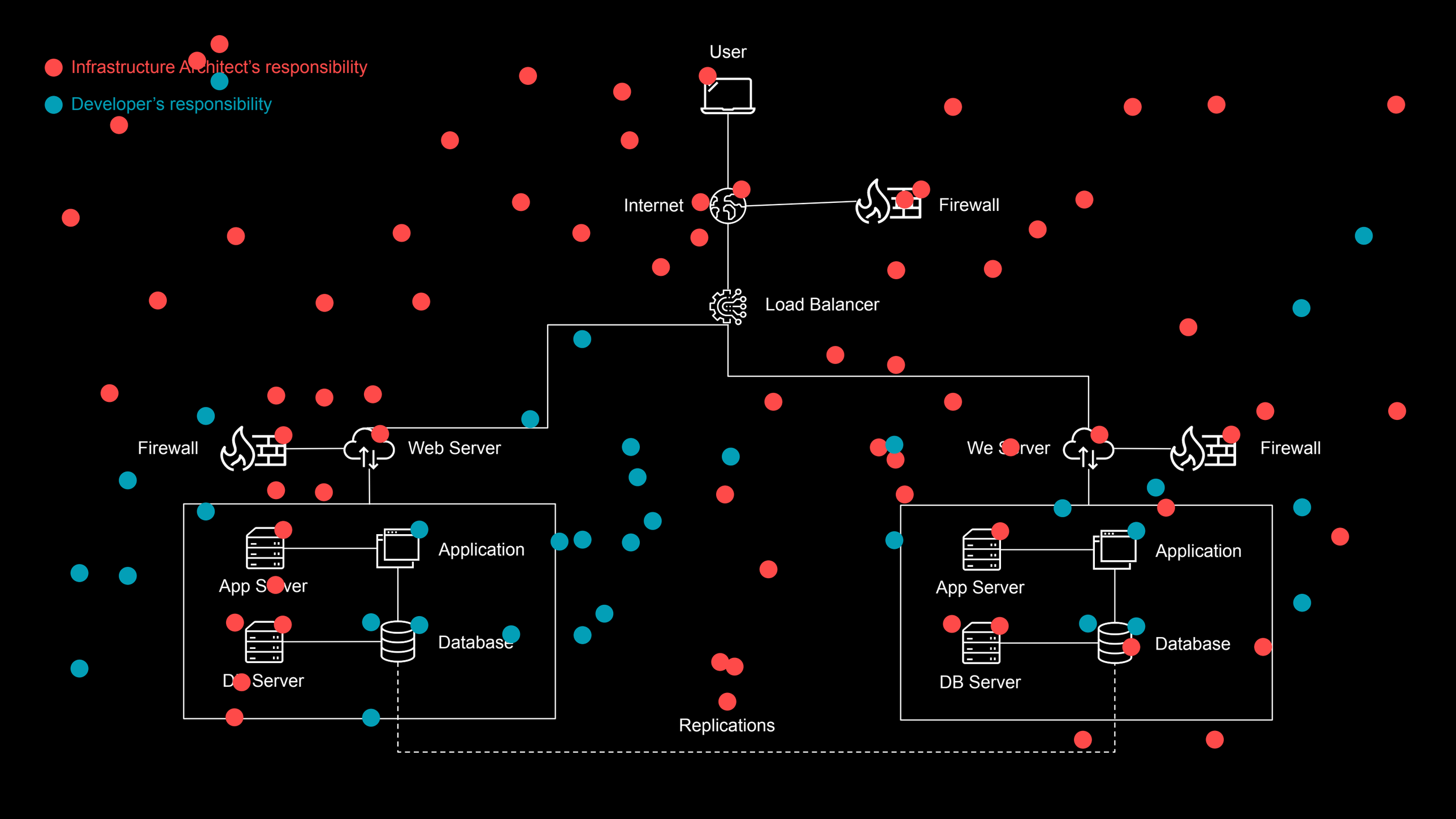

Infrastructure Architect's responsibility
User
Internet
Firewall
Load Balancer
Firewall
Web Server
We Server
Firewall
Application
App Server
Database
DB Server
Application
App Server
Database
DB Server
Replications


Infrastructure Architect's responsibility
User
Internet
Firewall
Load Balancer
Firewall
Web Server
We Server
Firewall
Application
App Server
Database
DB Server
Application
App Server
Database
DB Server
Replications


Infrastructure Architect's responsibility
User
Internet
Firewall
Load Balancer
Firewall
Web Server
We Server
Firewall
Application
App Server
Database
DB Server
Application
App Server
Database
DB Server
Replications


Infrastructure Architect's responsibility
User
Internet
Firewall
Load Balancer
Firewall
Web Server
We Server
Firewall
Application
App Server
Database
DB Server
Application
App Server
Database
DB Server
Replications

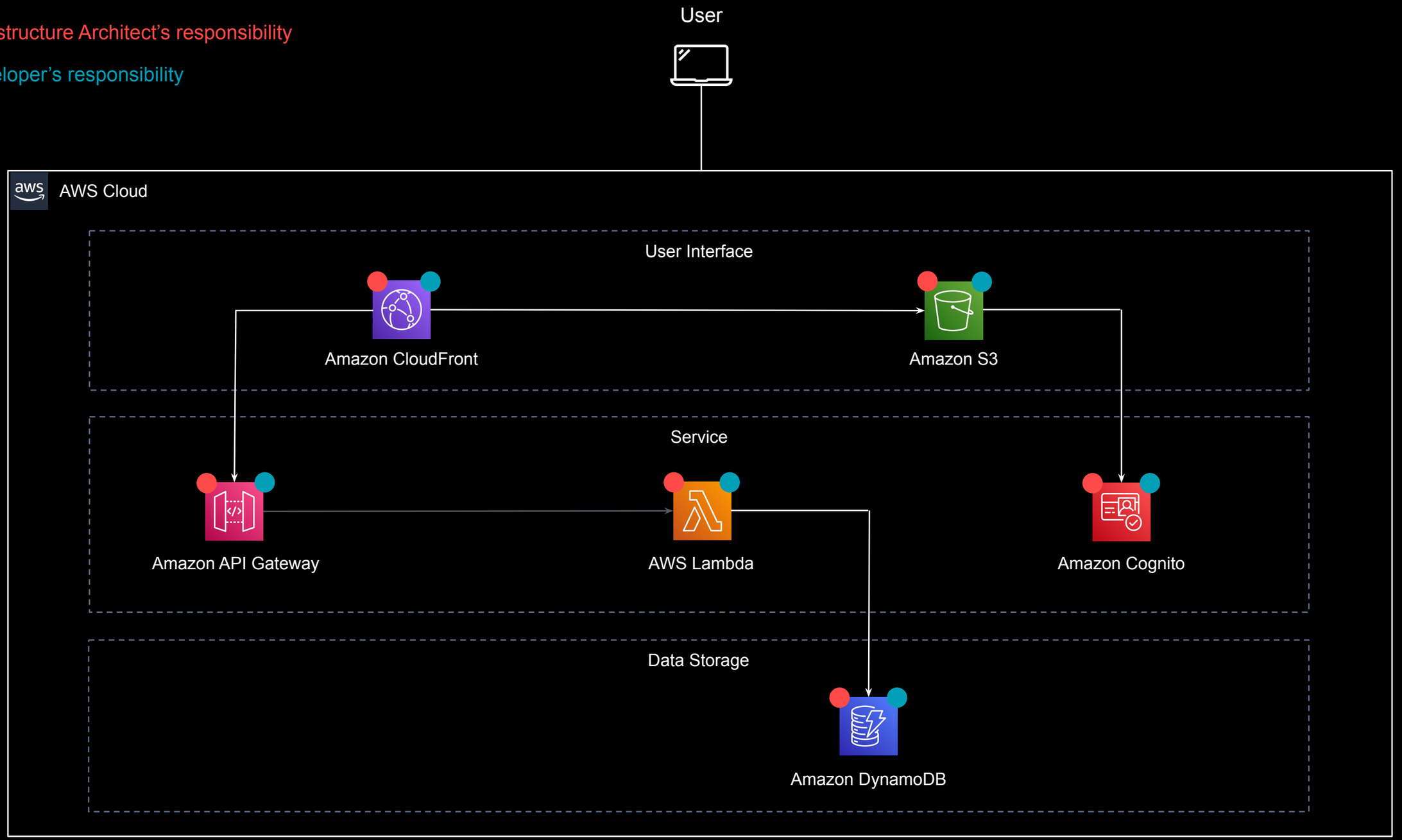

Infrastructure Architect's responsibility
Developer's responsibility
User
Internet
Firewall
Load Balancer
Firewall
Web Server
We Server
Firewall
Application
App Server
Database
DB Server
Application
App Server
Database
DB Server
Replications


Infrastructure Architect's responsibility
Developer's responsibility
User
Internet
Firewall
Load Balancer
Firewall
Web Server
Application
App Server
Database
DB Server
Replications
Application
App Server
Database
DB Server
Firewall

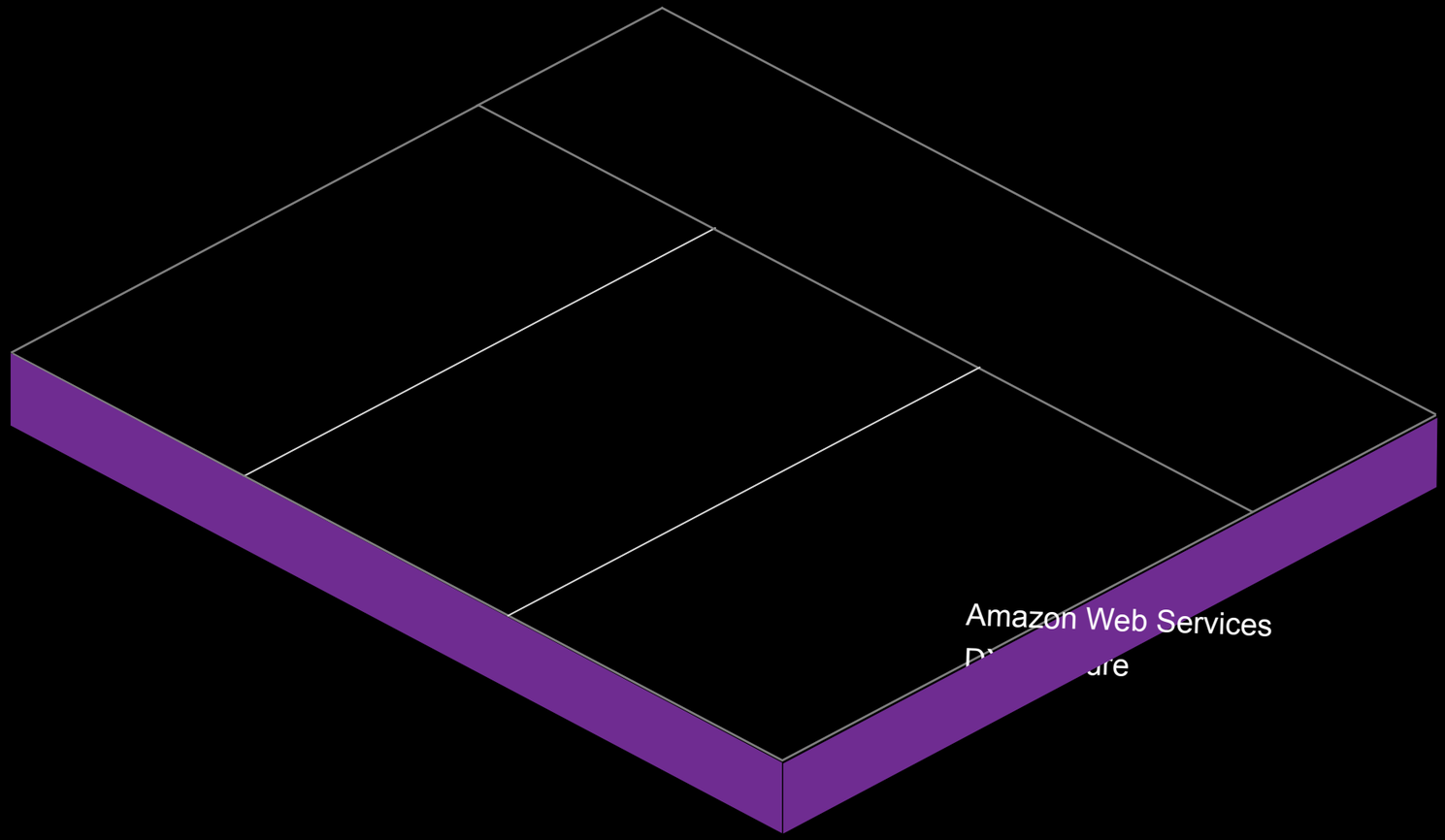
Picture by Anastasia Petrova via Unsplash

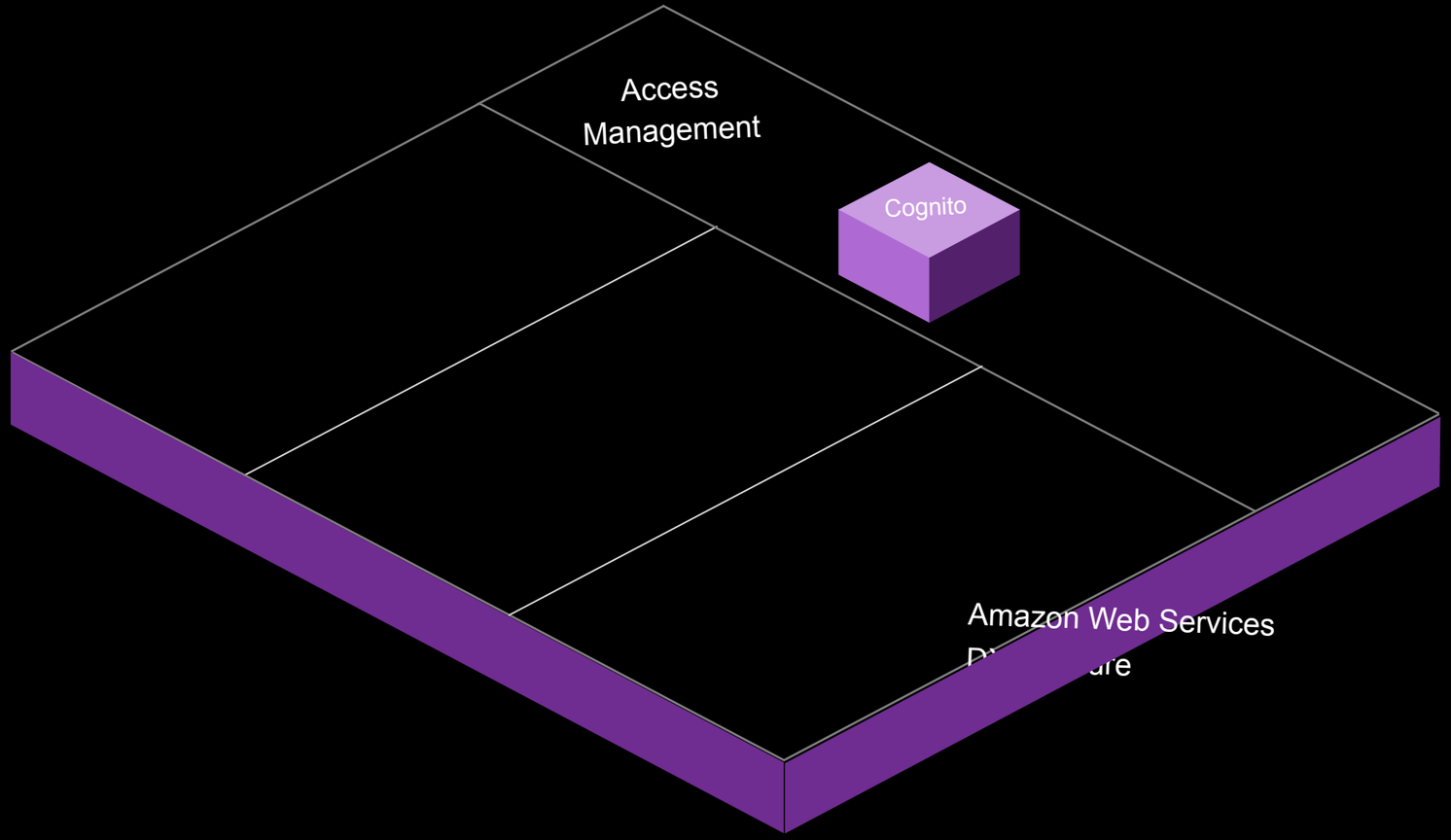
Infrastructure Architect's responsibility

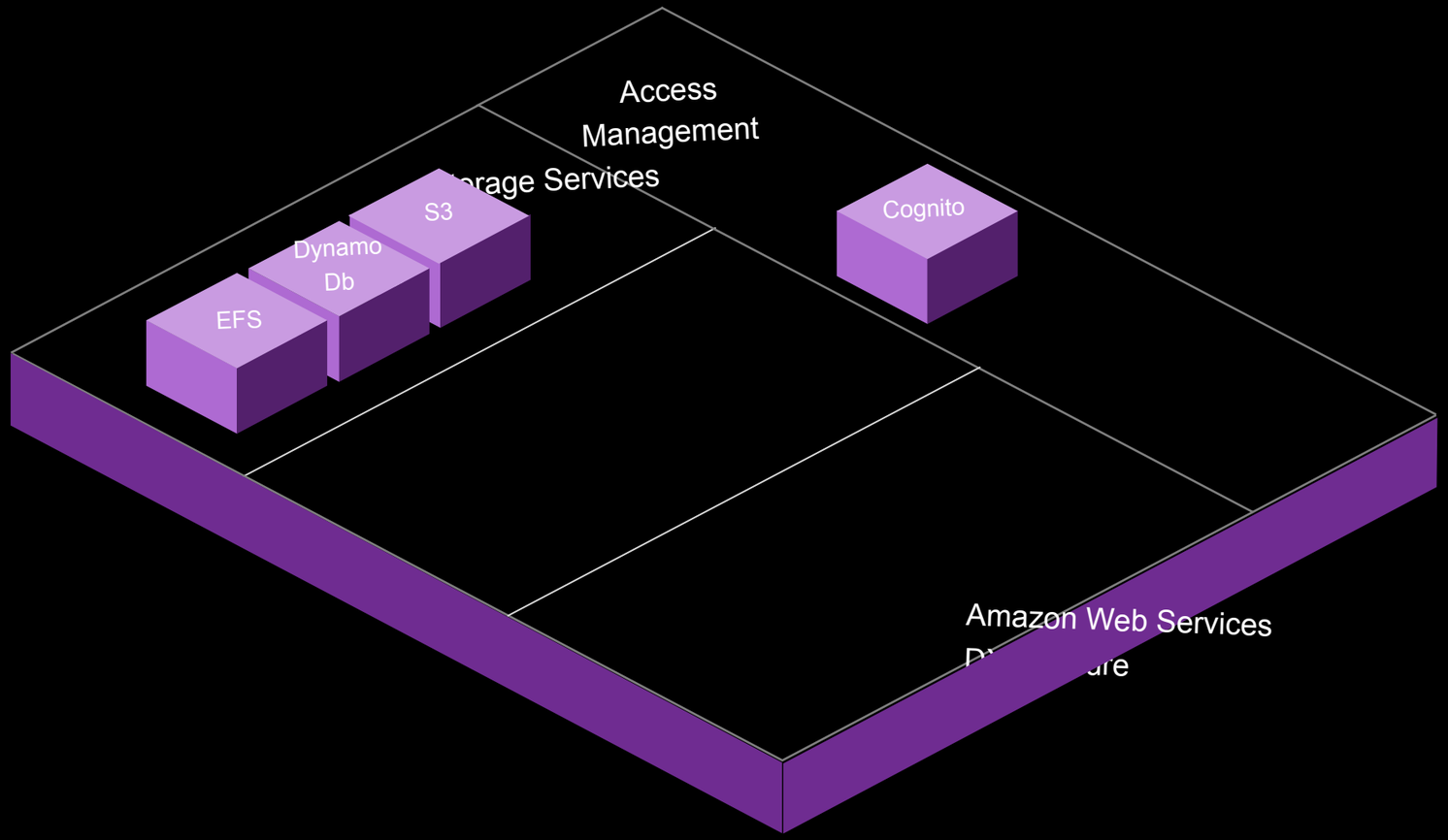
Developer's responsibility

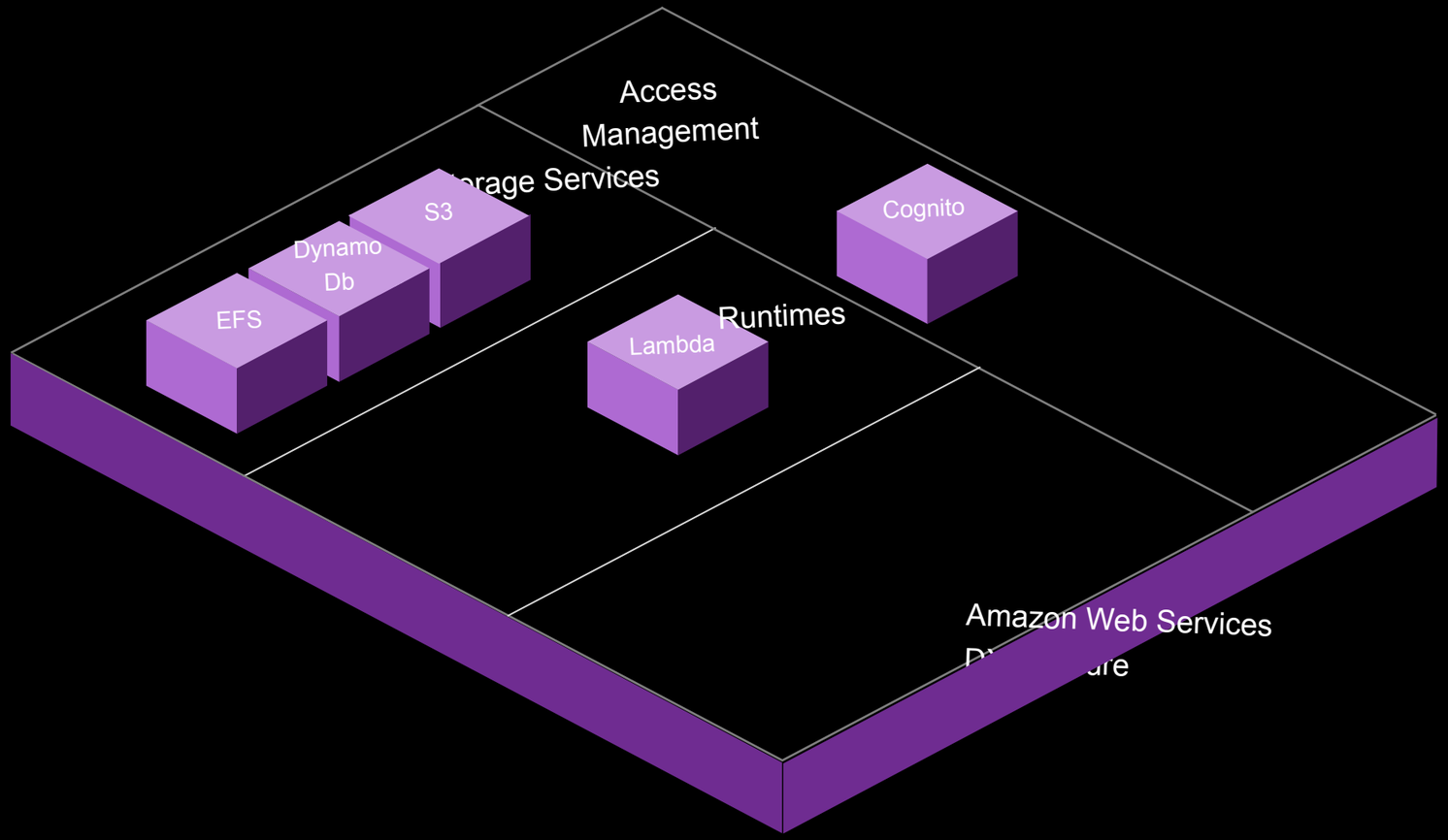


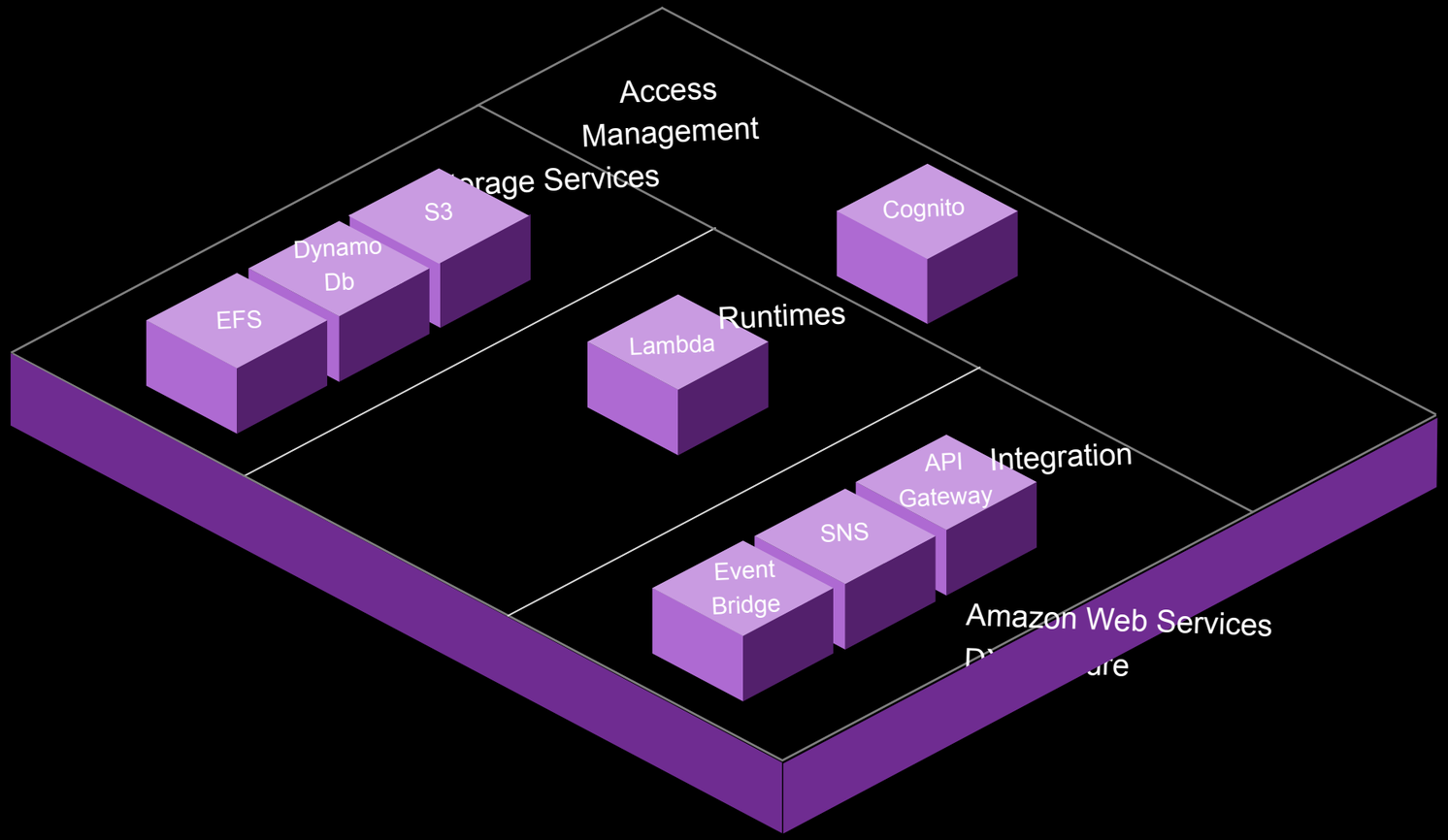
You are basically deploying your UML communication diagram

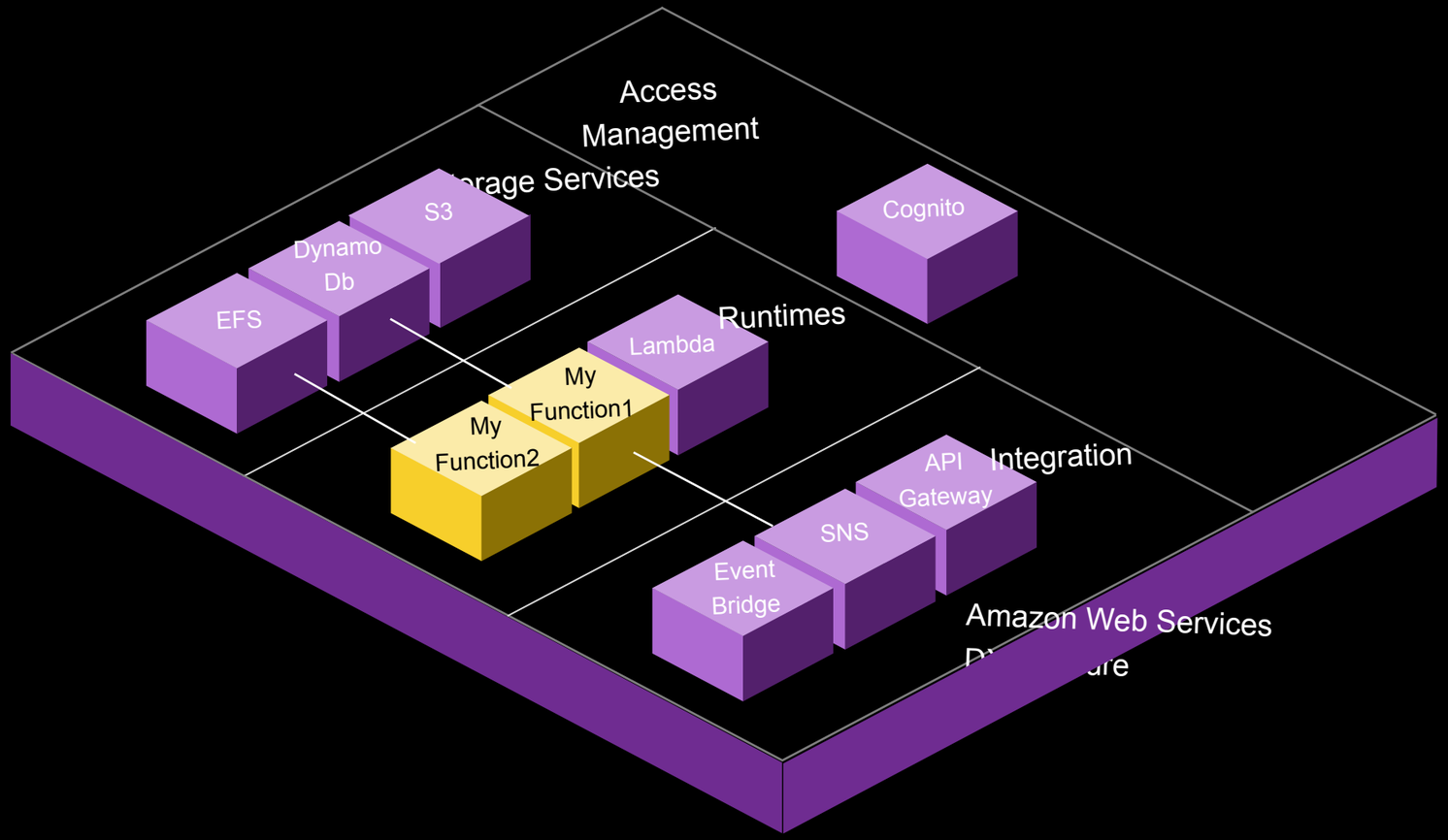

Amazon Web Services


Access
Management
Cognito
Amazon Web Services


Access
Management
S3
Dynamo
Db
EFS
Cognito
Amazon Web Services


Access
Management
S3
Dynamo
Db
EFS
Cognito
Runtimes
Lambda
Amazon Web Services


Access
Management
Cognito
S3
Dynamo
Db
EFS
Runtimes
Lambda
Integration
API
Gateway
SNS
Event
Bridge
Amazon Web Services

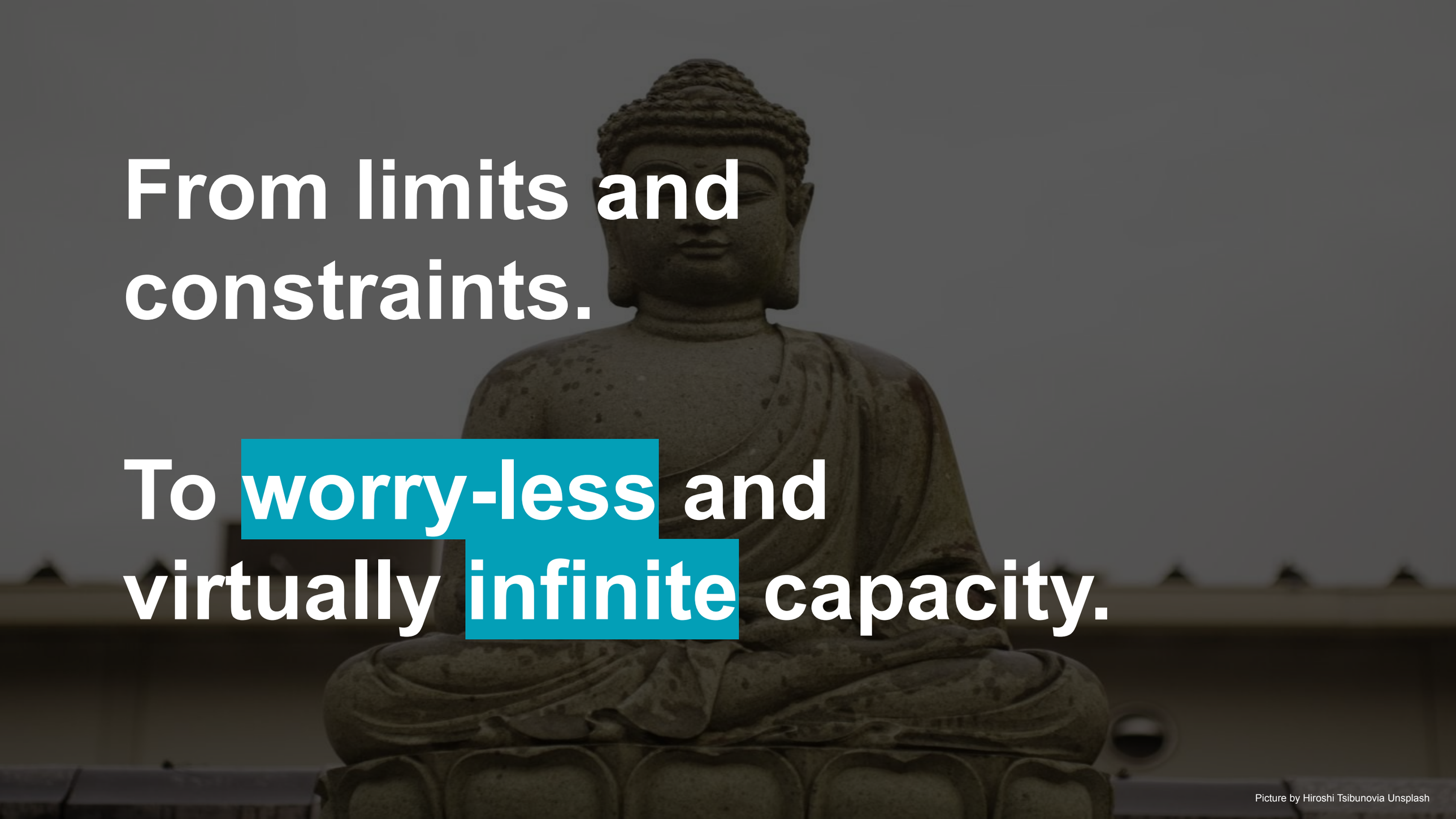


Managed cloud service

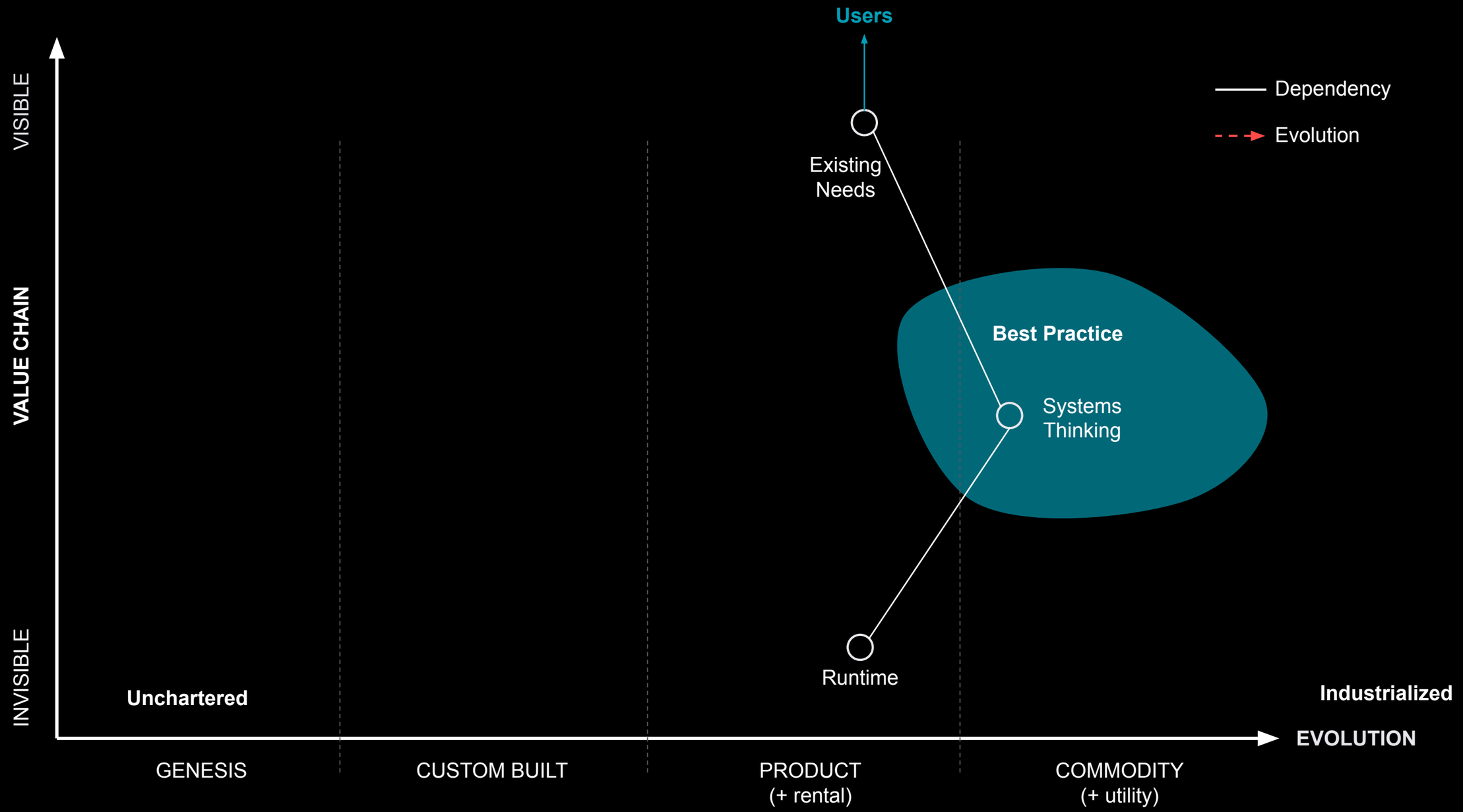
My application

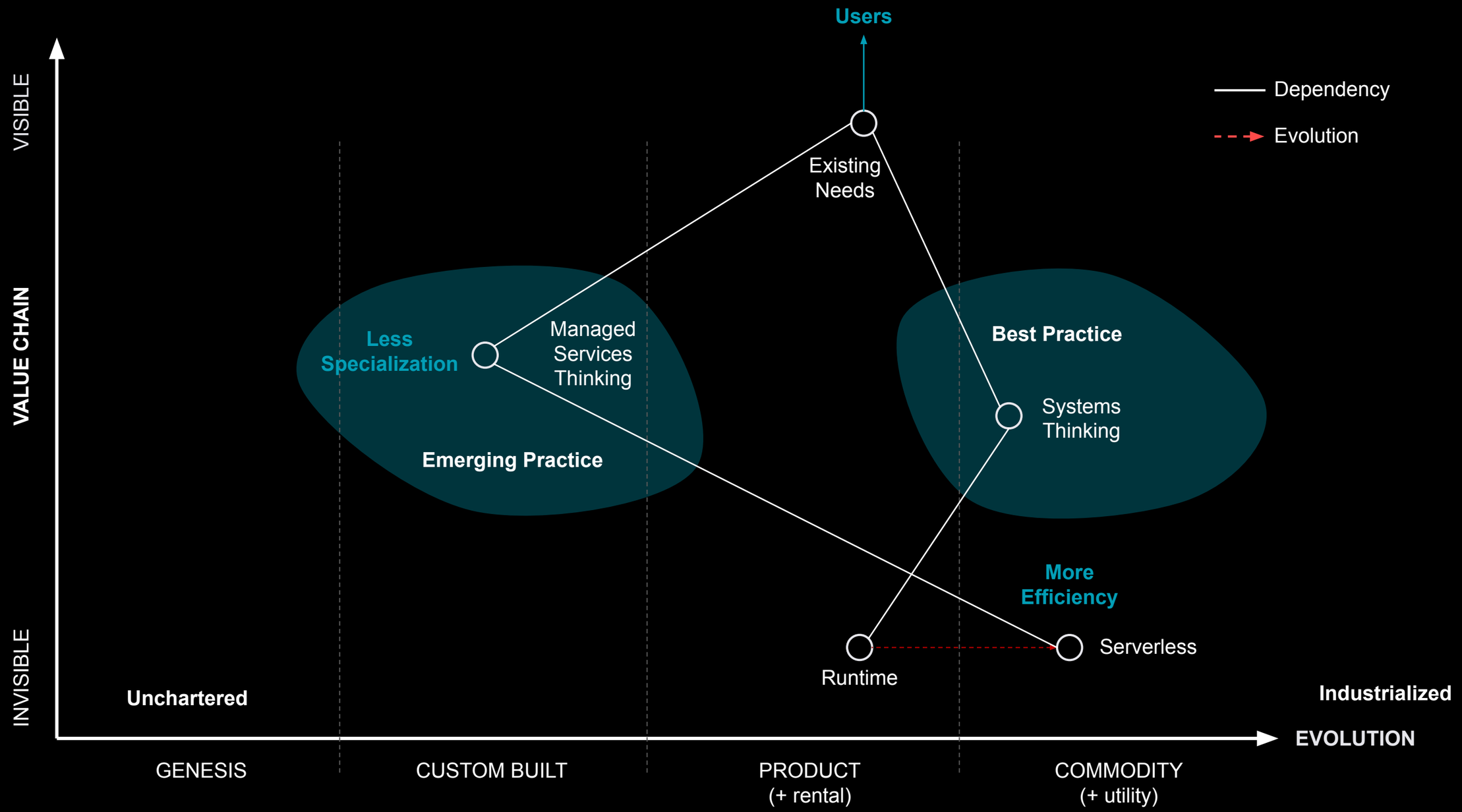
There is a shift for every software engineer.

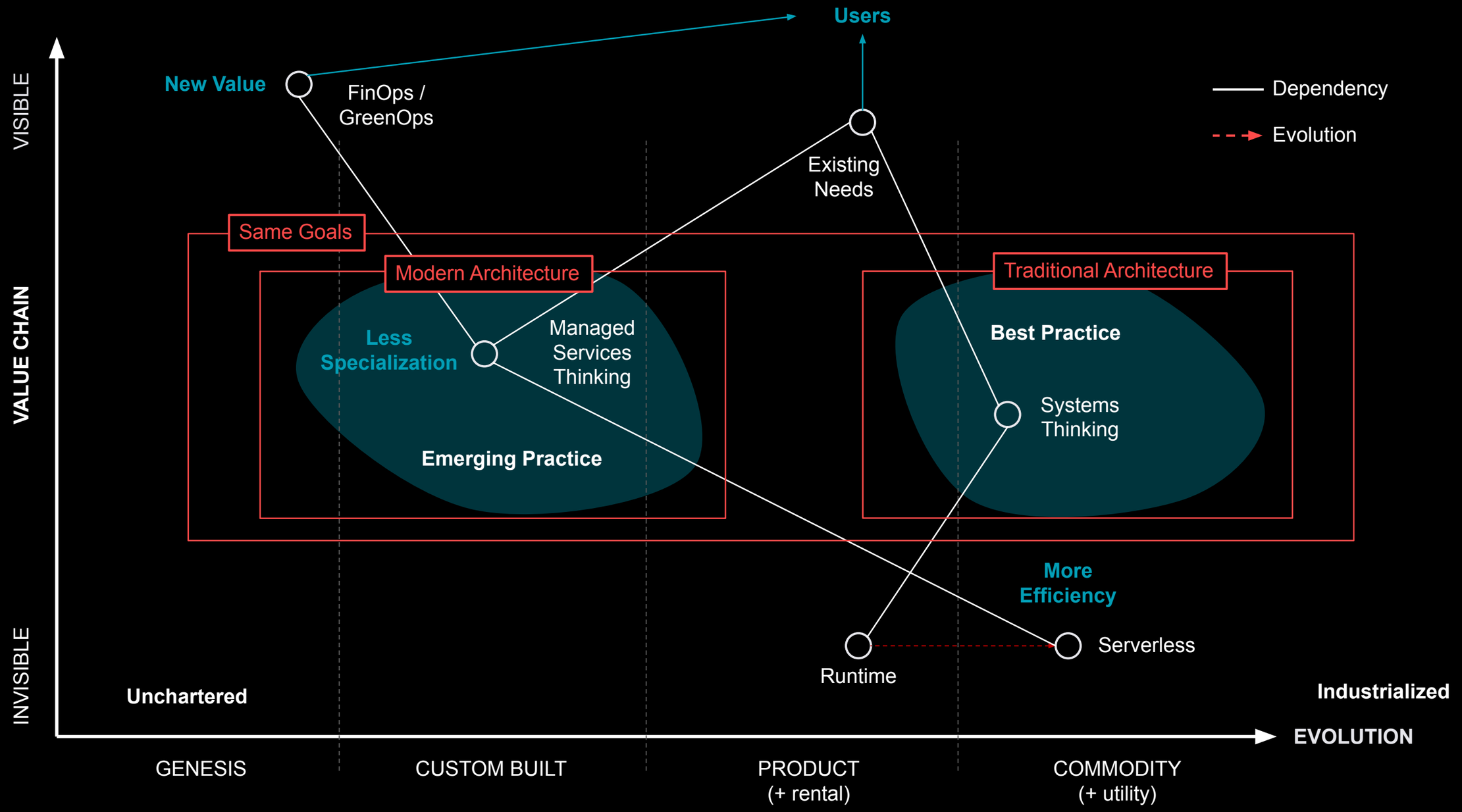
From systems thinking to managed-services thinking.

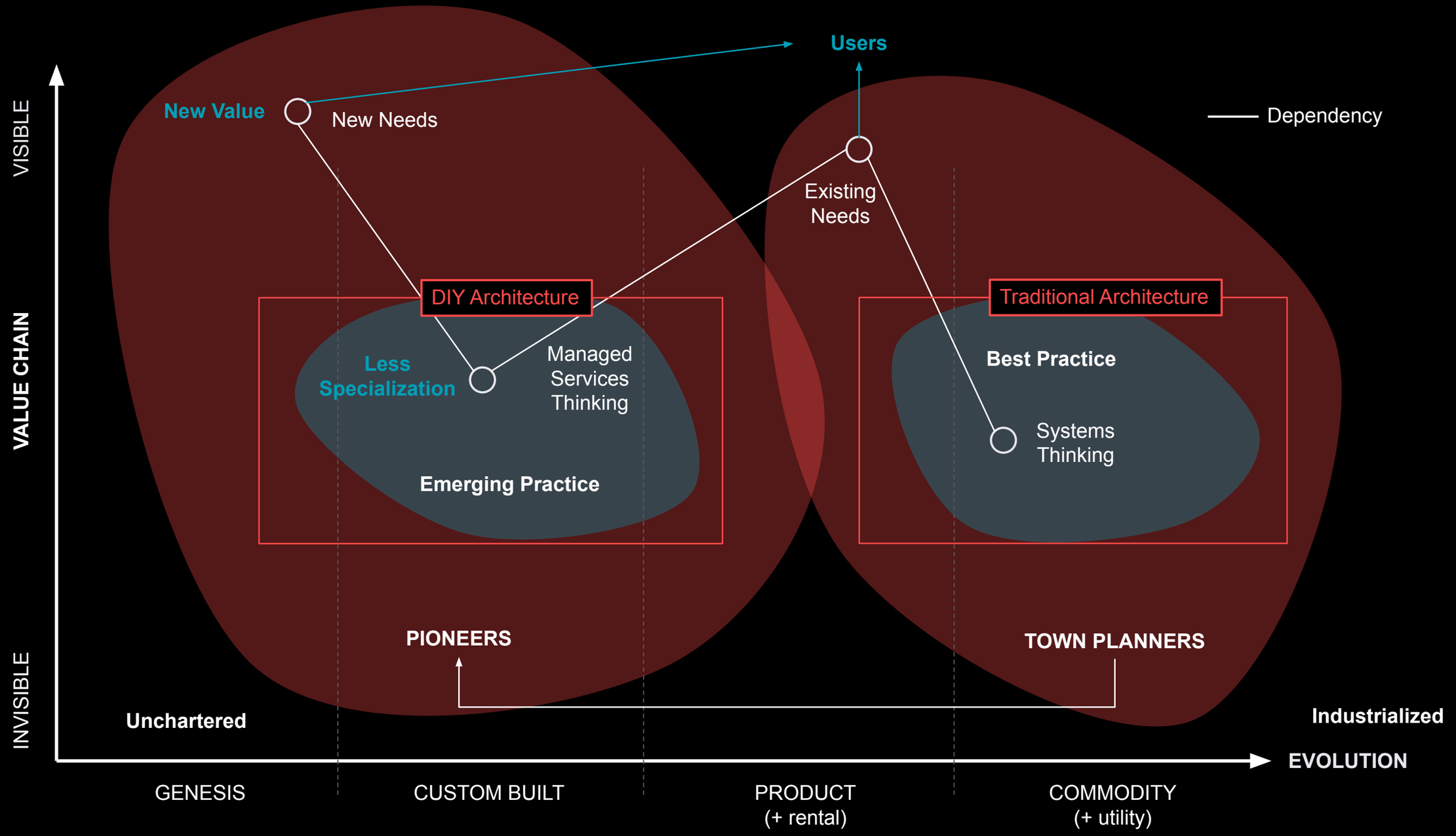


Picture by Hiroshi Tsibunovia Unsplash


Users
Dependency
Evolution
VISIBLE
VALUE CHAIN
INVISIBLE
Existing
Needs
Best Practice
Systems
Thinking
Runtime
Unchartered
Industrialized
EVOLUTION
GENESIS
CUSTOM BUILT
PRODUCT
(+ rental)
COMMODITY
(+ utility)


Users
Dependency
Evolution
Existing
Needs
Less
Specialization
Managed
Services
Thinking
Emerging Practice
Best Practice
Systems
Thinking
More
Efficiency
Serverless
Runtime
VISIBLE
VALUE CHAIN
INVISIBLE
Unchartered
Industrialized
EVOLUTION
GENESIS
CUSTOM BUILT
PRODUCT
(+ rental)
COMMODITY
(+ utility)


Users
New Value
FinOps /
GreenOps
Existing
Needs
Dependency
Evolution
Same Goals
Modern Architecture
Traditional Architecture
Less
Specialization
Managed
Services
Thinking
Emerging Practice
Best Practice
Systems
Thinking
More
Efficiency
Runtime
Serverless
VISIBLE
VALUE CHAIN
INVISIBLE
Uncharted
Industrialized
EVOLUTION
GENESIS
CUSTOM BUILT
PRODUCT
(+ rental)
COMMODITY
(+ utility)

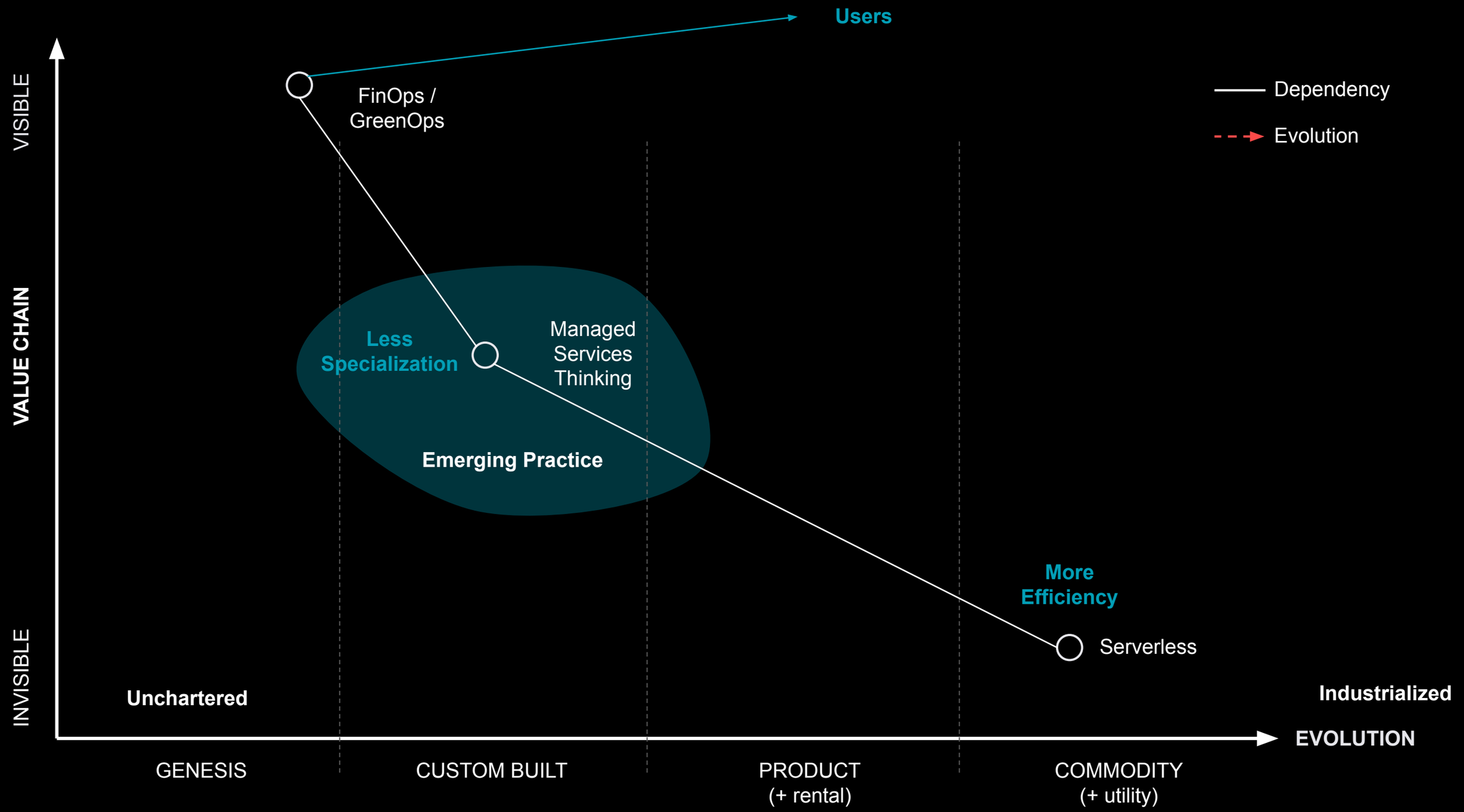

Users
New Value
New Needs
Dependency
Existing
Needs
DIY Architecture
Traditional Architecture
Less
Specialization
Managed
Services
Thinking
Emerging Practice
Best Practice
Systems
Thinking
PIONEERS
TOWN PLANNERS
VISIBLE
VALUE CHAIN
INVISIBLE
Uncharted
Industrialized
EVOLUTION
GENESIS
CUSTOM BUILT
PRODUCT
(+ rental)
COMMODITY
(+ utility)

New needs?

New value?

New NFRs, new practices.

We can now trace and measure the costs and environmental sustainability of our software design.

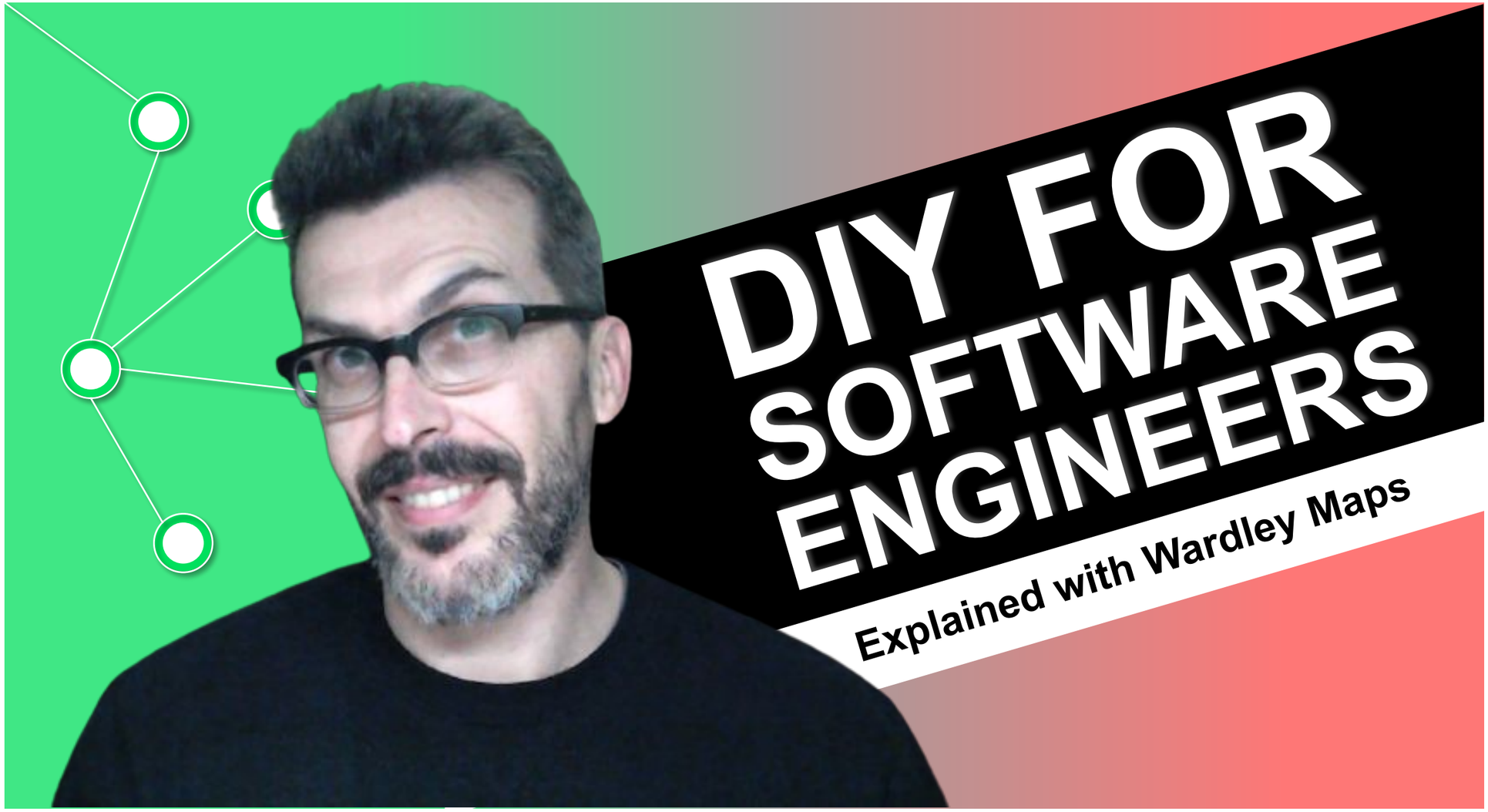

Users
FinOps /
GreenOps
Dependency
Evolution
VISIBLE
VALUE CHAIN
INVISIBLE
Less
Specialization
Managed
Services
Thinking
Emerging Practice
More
Efficiency
Serverless
Unchartered
Industrialized
EVOLUTION
GENESIS
CUSTOM BUILT
PRODUCT
(+ rental)
COMMODITY
(+ utility)

Beware the micro-optimizations.

We shouldn’t trade these new NFRs for other equally important ones such as code maintainability.

Thank you!


Building
Software Platforms
A Guide to SaaS Transition with AWS
Pablo Bermejo
With foreword by Bill Ohnemus


DIY FOR
SOFTWARE
ENGINEERS
Explained with Wardley Maps

🤏 Ilike
this