Liferay Portal 7
(micro)services for the enterprise

Milen Dyankov
@milendyankov

martinfowler.com/articles/microservices.html

MARTIN FOWLER

Intro Design Agile Refactoring NoSQL DSL Delivery About Me ThoughtWorks

Microservices

The term "Microservice Architecture" has sprung up over the last few years to describe a particular way of designing software applications as suites of independently deployable services. While there is no precise definition of this architectural style, there are certain common characteristics around organization around business capability, automated deployment, intelligence in the endpoints, and decentralized control of languages and data.

25 March 2014

James Lewis

James Lewis is a Principal Consultant at ThoughtWorks and member of the Technology Advisory Board. James' interest in building applications out of small collaborating services stems from a background in integrating enterprise systems at scale. He's built a number of systems using microservices and has been an active participant in the growing community for a couple of years.

Martin Fowler

Martin Fowler is an author, speaker, and general loud-mouth on software development. He's long been puzzled by the problem of how to componentize software systems, having heard more vague claims than he's happy with. He hopes that microservices will live up to the early promise its advocates have found.

Translations: Japanese · Russian ·

Find **similar articles** to this by looking at these tags: popular · application architecture · web services · microservices

Contents

Characteristics of a Microservice Architecture
- Componentization via Services
- Organized around Business Capabilities
- Products not Projects
- Smart endpoints and dumb pipes
- Decentralized Governance
- Decentralized Data Management
- Infrastructure Automation
- Design for failure
- Evolutionary Design

Are Microservices the Future?

Sidebars

How big is a microservice?
Microservices and SOA
Many languages, many options
Battle-tested standards and enforced standards
Make it easy to do the right thing
The circuit breaker and production ready code
Synchronous calls considered harmful

Microservices characteristics!

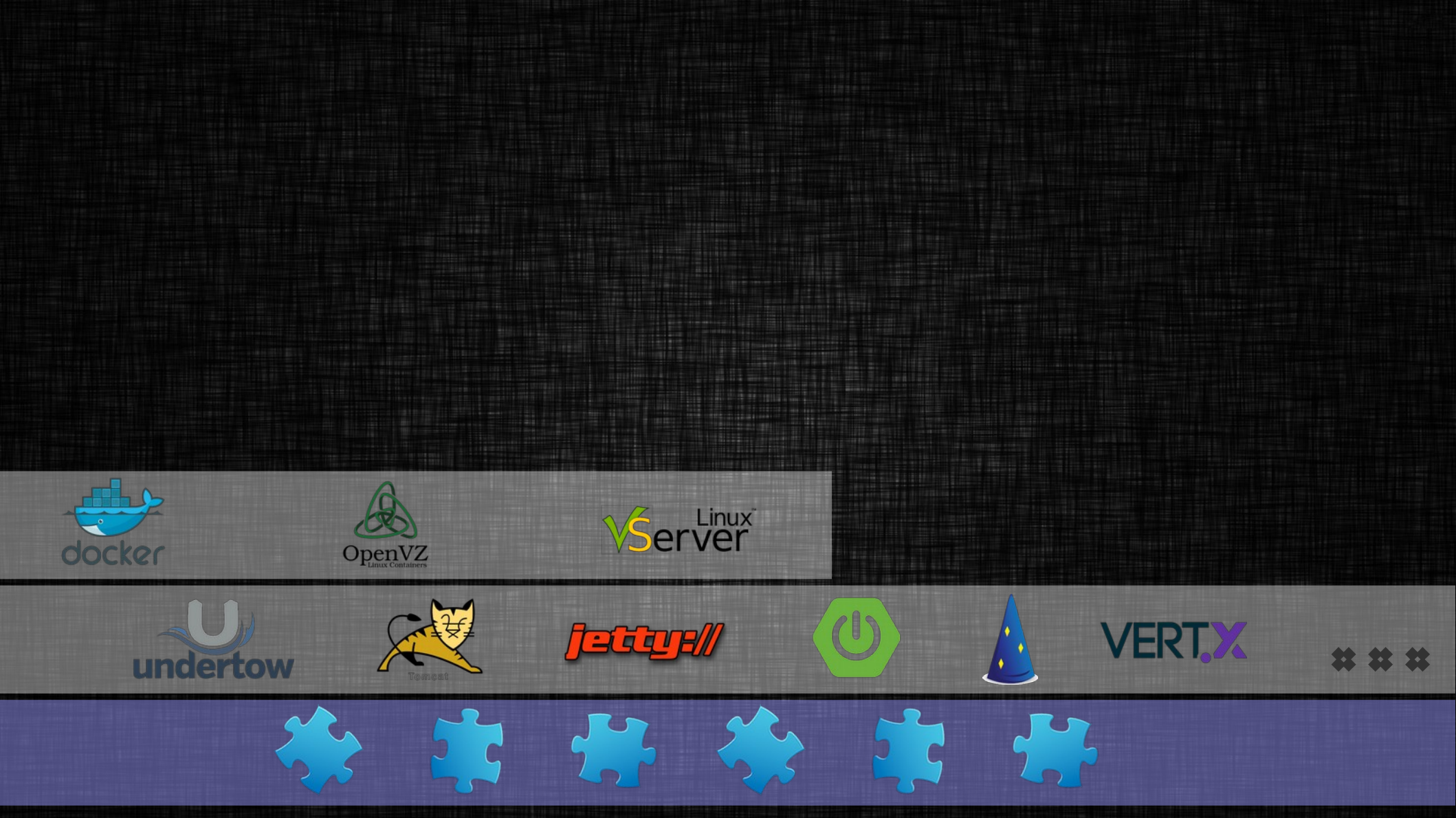
- Componentization via Services
- Organized around Business Capabilities
- Products not Projects
- Smart endpoints and dumb pipes
- Decentralized Governance
- Decentralized Data Management
- Infrastructure Automation
- Design for failure
- Evolutionary Design

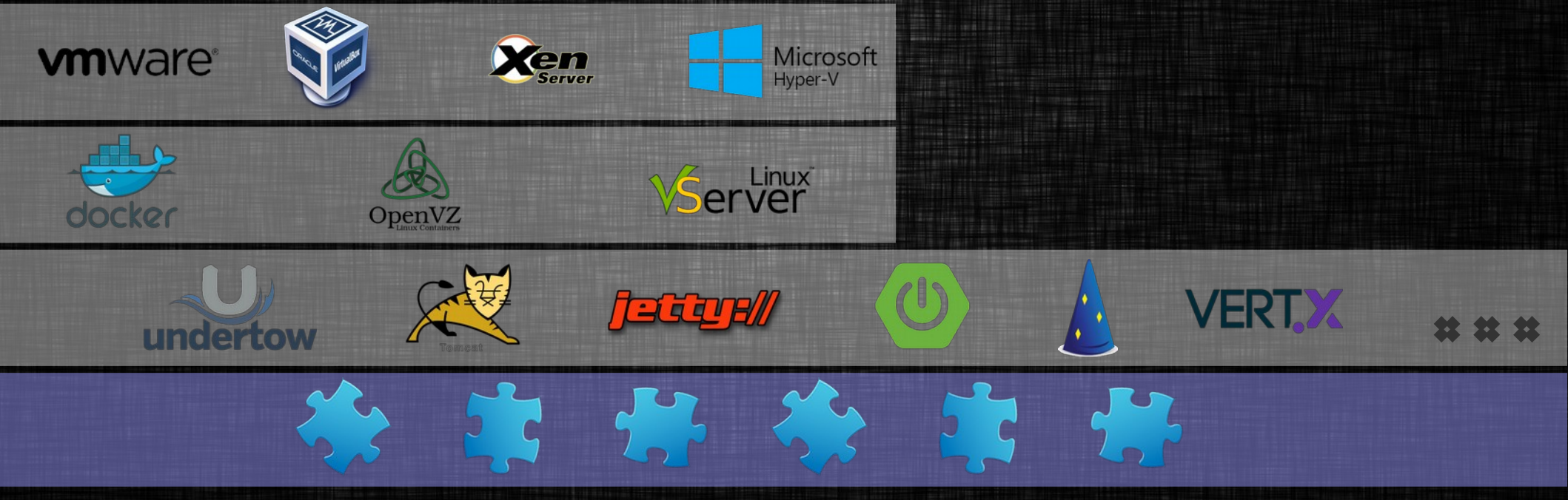
undertow

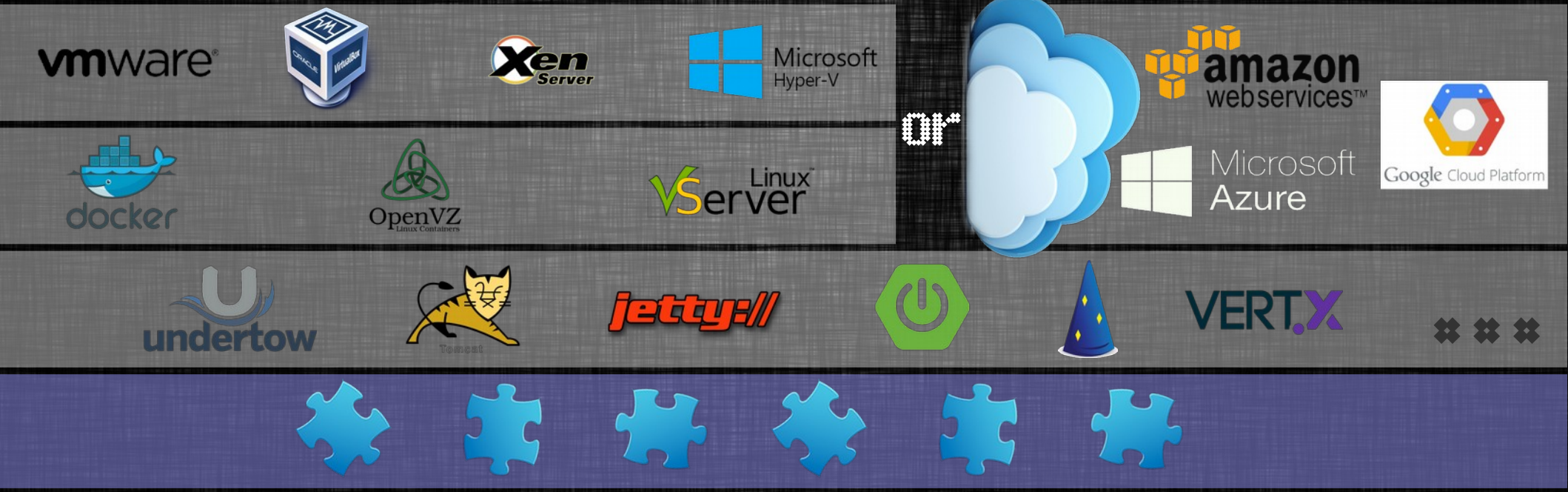
Tomcat

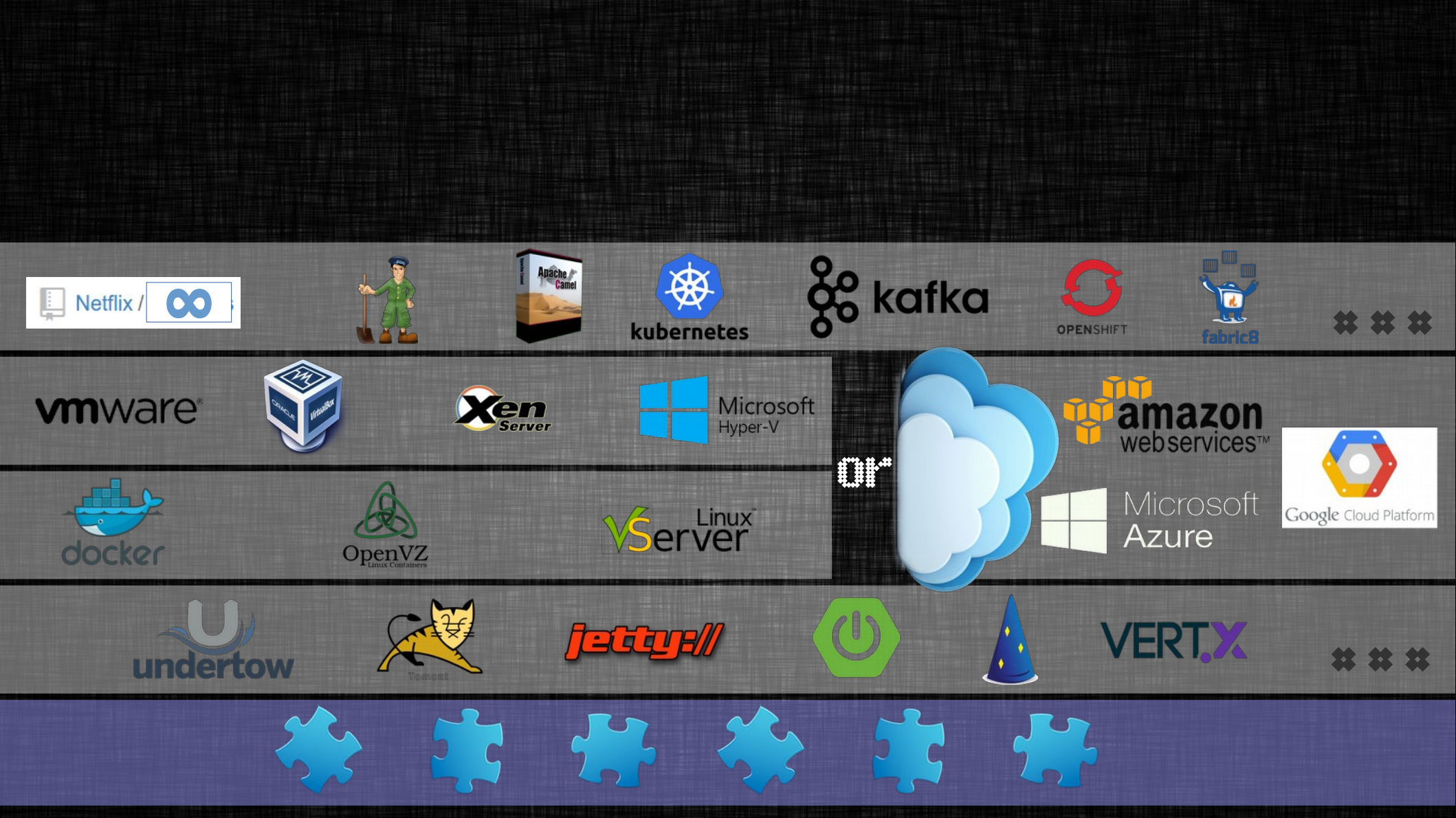
jetty://

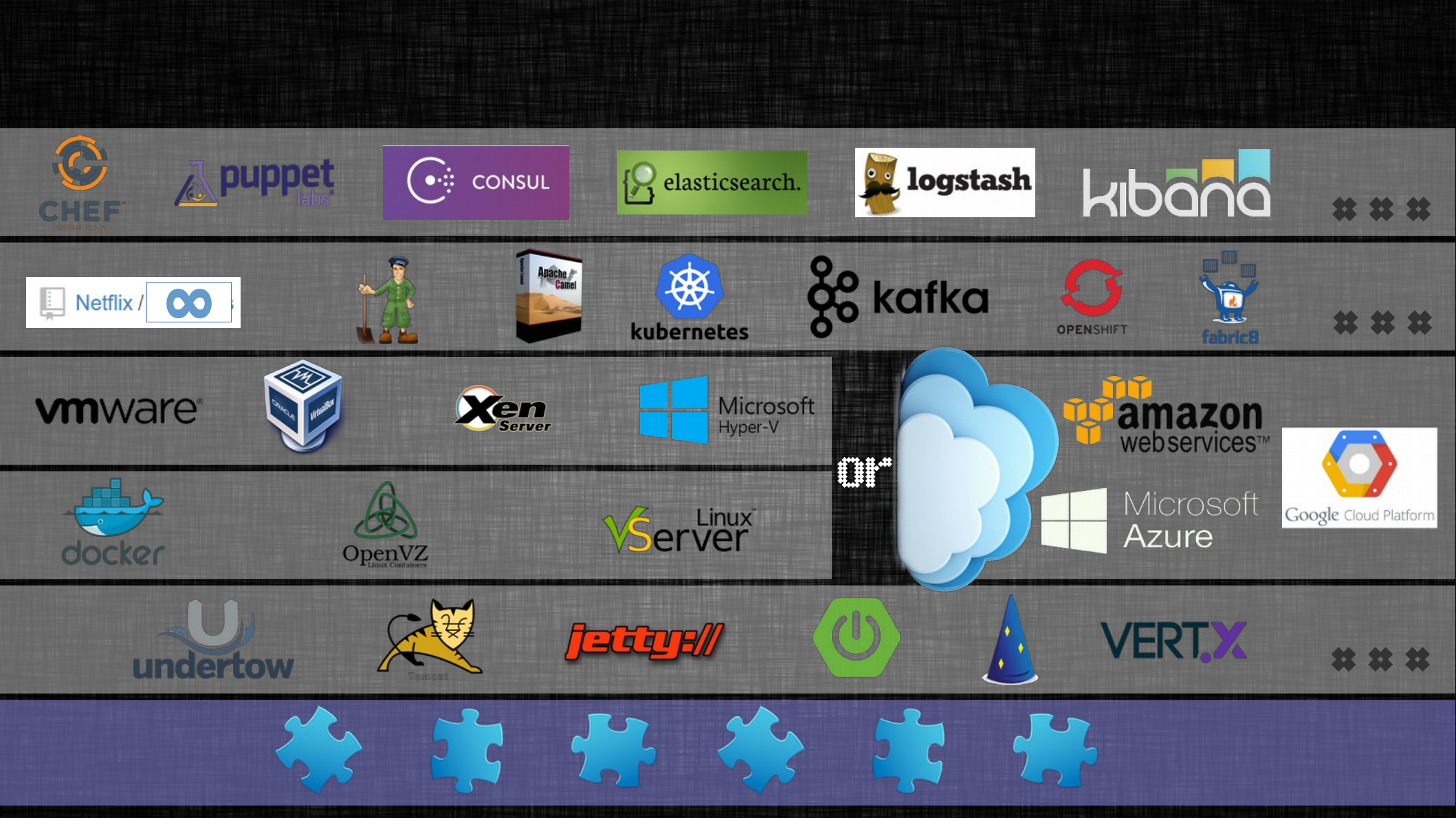
VERT.X


docker
OpenVZ
Linux Containers
Linux
VServer
undertow
Tomcat
jetty://
VERT.X

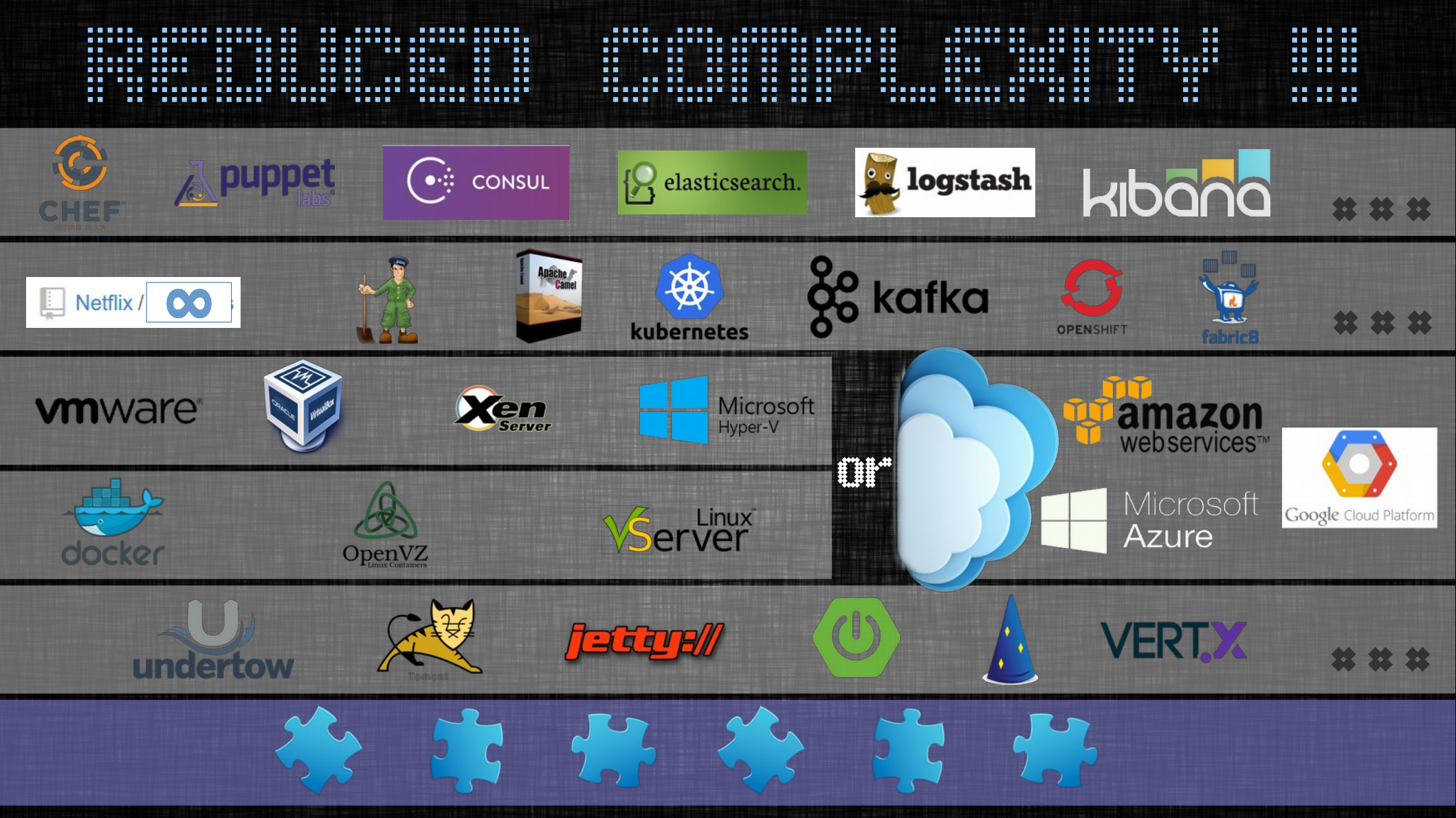

vmware
Xen Server
Microsoft Hyper-V
docker
OpenVZ
Linux VServer
undertow
Tomcat
jetty://
VERT.X


vmware
Xen Server
Microsoft Hyper-V
or
amazon web services™
Google Cloud Platform
Microsoft Azure
docker
OpenVZ Linux Containers
Linux VServer
undertow
Tomcat
jetty://
VERT.X


Netflix /
Apache Camel
kubernetes
kafka
OPENSHIFT
fabric8
vmware
Xen Server
Microsoft Hyper-V
or
amazon web services™
Google Cloud Platform
Microsoft Azure
docker
OpenVZ Linux Containers
Linux VServer
undertow
Tomcat
jetty://
VERT.X


CHEF
puppet labs
CONSUL
elasticsearch.
logstash
kibana
Netflix /
Apache Camel
kubernetes
kafka
OPENSHIFT
fabric8
vmware
Xen Server
Microsoft Hyper-V
or
amazon web services
Google Cloud Platform
Microsoft Azure
docker
OpenVZ
Linux VServer
undertow
Tomcat
jetty://
VERT.X

REDUCED COMPLEXITY



martinfowler.com/articles/microservices.html

Often the true consequences of your architectural decisions are only evident several years after you made them. We have seen projects where a good team, with a strong desire for modularity, has built a monolithic architecture that has decayed over the years. Many people believe that such decay is less likely with microservices, since the service boundaries are explicit and hard to patch around. Yet until we see enough systems with enough age, we can't truly assess how microservice architectures mature.

There are certainly reasons why one might expect microservices to mature poorly. In any effort at componentization, success depends on how well the software fits into components. It's hard to figure out exactly where the component boundaries should lie. Evolutionary design recognizes the difficulties of getting boundaries right and thus the importance of it being easy to refactor them. But when your components are services with remote communications, then refactoring is much harder than with in-process libraries. Moving code is difficult across service boundaries, any interface changes need to be coordinated between participants, layers of backwards compatibility need to be added, and testing is made more complicated.

Another issue is If the components do not compose cleanly, then all you are doing is shifting complexity from inside a component to the connections between components. Not just does this just move complexity around, it moves it to a place that's less explicit and harder to control. It's easy to think things are better when you are looking at the inside of a small, simple component, while missing messy connections between services.

Finally, there is the factor of team skill. New techniques tend to be adopted by more skillful teams. But a technique that is more effective for a more skillful team isn't necessarily going to work for less skillful teams. We've seen plenty of cases of less skillful teams building messy monolithic architectures, but it takes time to see what happens when this kind of mess occurs with microservices. A poor team will always create a poor system - it's very hard to tell if microservices reduce the mess in this case or make it worse.

One reasonable argument we've heard is that you shouldn't start with a microservices architecture. Instead begin with a monolith, keep it modular, and split it into microservices once the monolith becomes a problem. (Although this advice isn't ideal, since a good in-process interface is usually not a good service interface.)

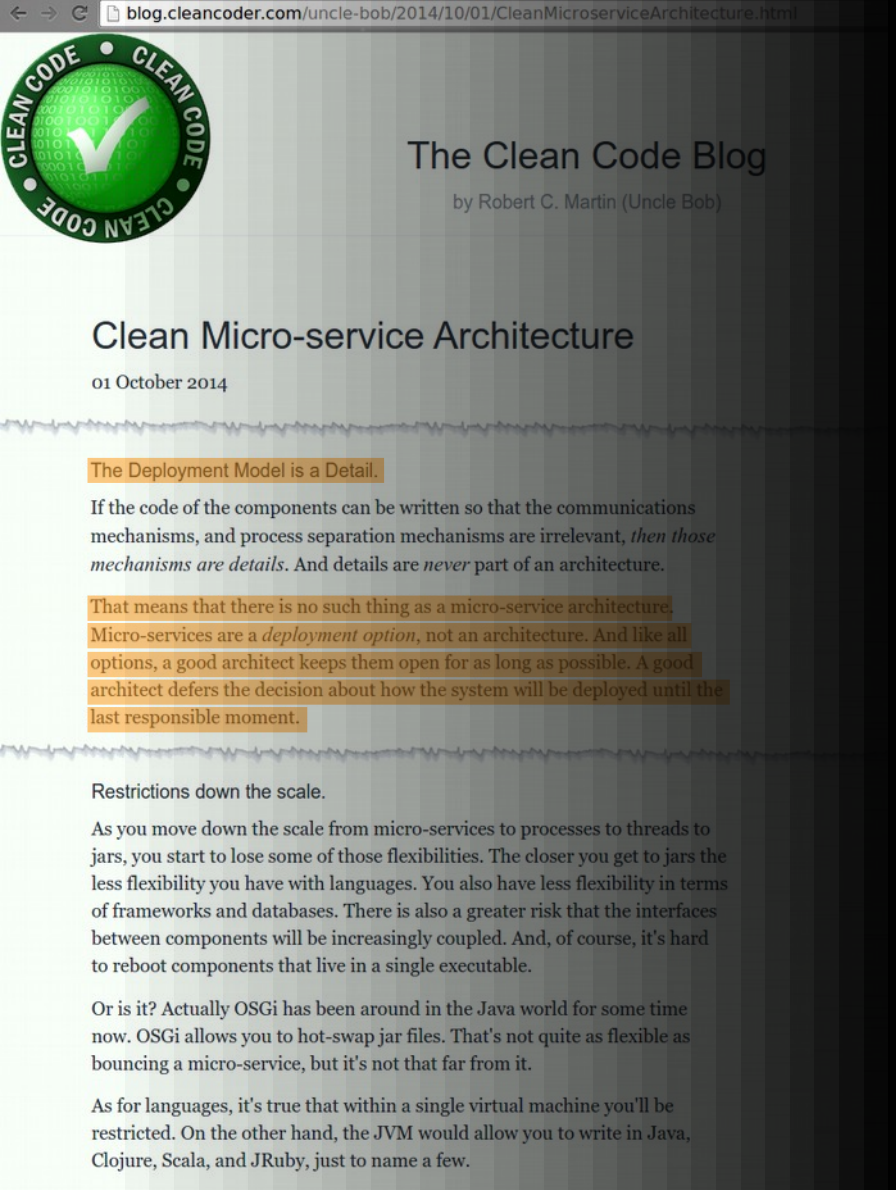
So we write this with cautious optimism. So far, we've seen enough about the microservice style to feel that it can be a worthwhile road to tread. We can't say for sure where we'll end up, but one of the challenges of software development is that you can only make decisions based on the imperfect information that you currently have to hand.

. . .

then all you are doing is **shifting complexity** from inside a component to the connections between components.

. . .

it moves it **to a place that's less explicit and harder to control.**



. . .

The Deployment Model is a Detail.

. . .

there is no such thing as a micro-service architecture.

Micro-services are a deployment option

. . .

martinfowler.com/articles/microservices.html

MARTIN FOWLER

Intro Design Agile Refactoring NoSQL DSL Delivery About Me ThoughtWorks

Microservices

The term "Microservice Architecture" has sprung up over the last few years to describe a particular way of designing software applications as suites of independently deployable services. While there is no precise definition of this architectural style, there are certain common characteristics around organization around business capability, automated deployment, intelligence in the endpoints, and decentralized control of languages and data.

25 March 2014

James Lewis

James Lewis is a Principal Consultant at ThoughtWorks and member of the Technology Advisory Board. James' interest in building applications out of small collaborating services stems from a background in integrating enterprise systems at scale. He's built a number of systems using microservices and has been an active participant in the growing community for a couple of years.

Martin Fowler

Martin Fowler is an author, speaker, and general loud-mouth on software development. He's long been puzzled by the problem of how to componentize software systems, having heard more vague claims than he's happy with. He hopes that microservices will live up to the early promise its advocates have found.

Translations: Japanese · Russian ·

Find **similar articles** to this by looking at these tags: popular · application architecture · web services · microservices

Contents

Characteristics of a Microservice Architecture
- Componentization via Services
- Organized around Business Capabilities
- Products not Projects
- Smart endpoints and dumb pipes
- Decentralized Governance
- Decentralized Data Management
- Infrastructure Automation
- Design for failure
- Evolutionary Design

Are Microservices the Future?

Sidebars

How big is a microservice?
Microservices and SOA
Many languages, many options
Battle-tested standards and enforced standards
Make it easy to do the right thing
The circuit breaker and production ready code
Synchronous calls considered harmful

50% not strictly software but rather operations related!

- Componentization via Services
- Organized around Business Capabilities
- Products not Projects
- Smart endpoints and dumb pipes
- Decentralized Governance
- Decentralized Data Management
- Infrastructure Automation
- Design for failure
- Evolutionary Design

martinfowler.com/articles/microservices.html

MARTIN FOWLER

Intro Design Agile Refactoring NoSQL DSL Delivery About Me ThoughtWorks

Microservices

The term "Microservice Architecture" has sprung up over the last few years to describe a particular way of designing software applications as suites of independently deployable services. While there is no precise definition of this architectural style, there are certain common characteristics around organization around business capability, automated deployment, intelligence in the endpoints, and decentralized control of languages and data.

25 March 2014

James Lewis

James Lewis is a Principal Consultant at ThoughtWorks and member of the Technology Advisory Board. James' interest in building applications out of small collaborating services stems from a background in integrating enterprise systems at scale. He's built a number of systems using microservices and has been an active participant in the growing community for a couple of years.

Martin Fowler

Martin Fowler is an author, speaker, and general loud-mouth on software development. He's long been puzzled by the problem of how to componentize software systems, having heard more vague claims than he's happy with. He hopes that microservices will live up to the early promise its advocates have found.

Translations: Japanese · Russian ·

Find **similar articles** to this by looking at these tags: popular · application architecture · web services · microservices

Contents

Characteristics of a Microservice Architecture
- Componentization via Services
- Organized around Business Capabilities
- Products not Projects
- Smart endpoints and dumb pipes
- Decentralized Governance
- Decentralized Data Management
- Infrastructure Automation
- Design for failure
- Evolutionary Design

Are Microservices the Future?

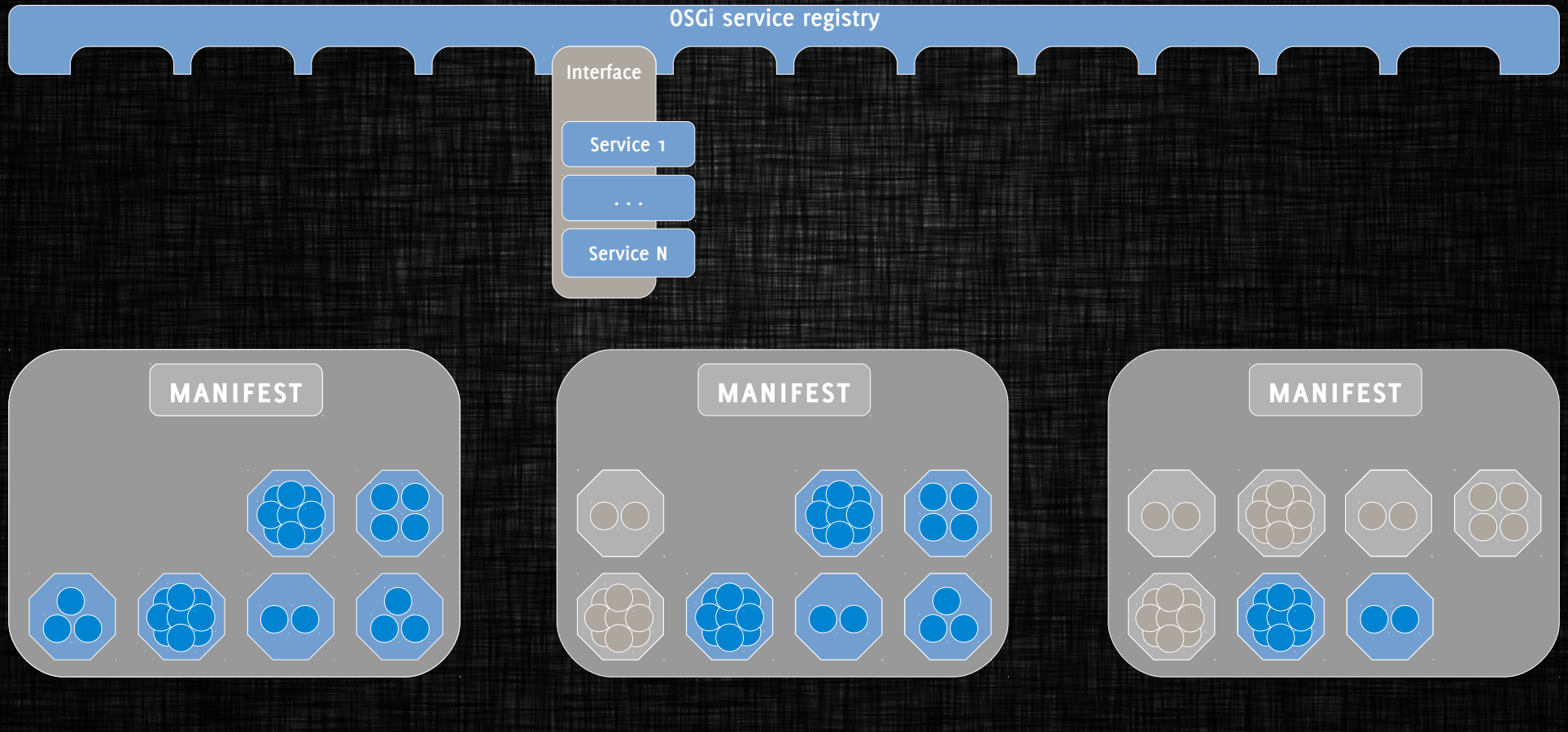
Sidebars

How big is a microservice?
Microservices and SOA
Many languages, many options
Battle-tested standards and enforced standards
Make it easy to do the right thing
The circuit breaker and production ready code
Synchronous calls considered harmful

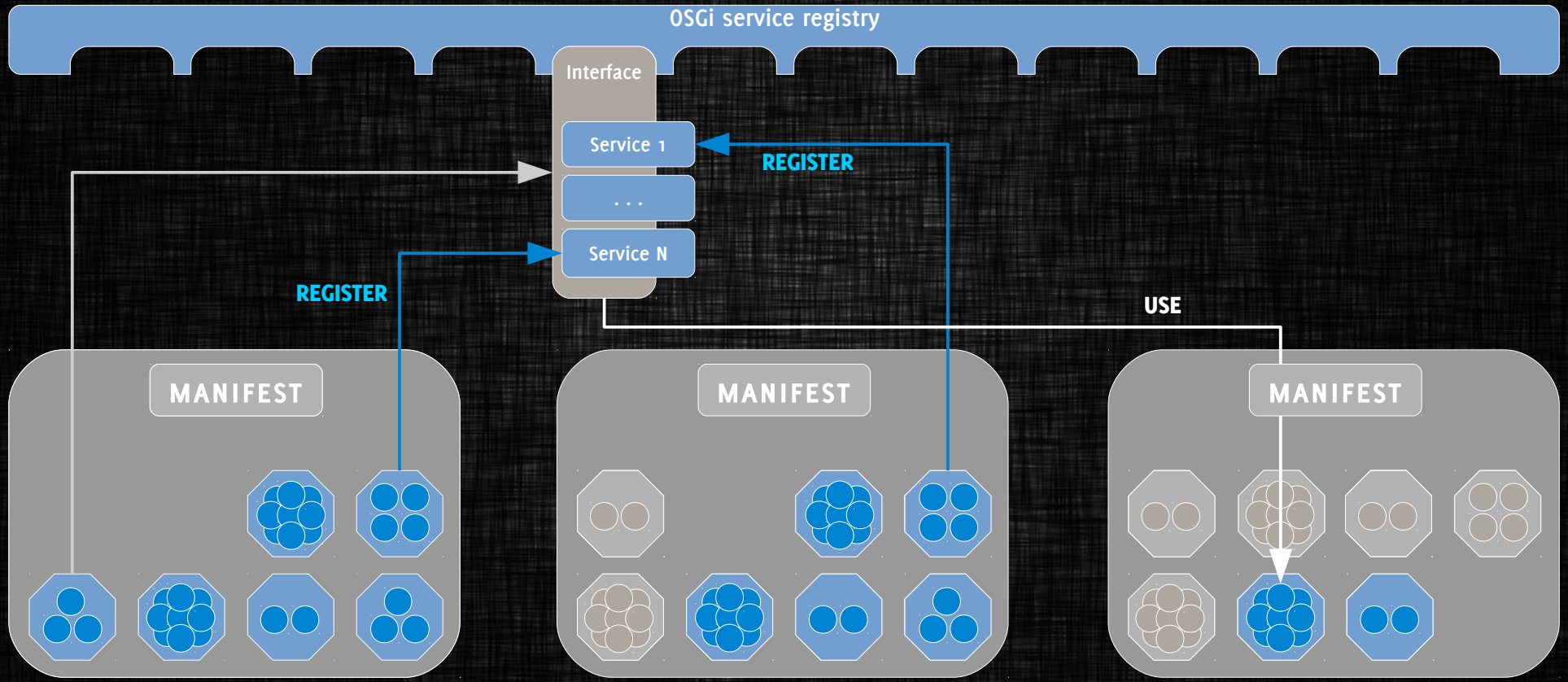
Let's focus on software related ones

- Componentization via Services
- Smart endpoints and dumb pipes
- Decentralized Data Management
- Design for failure

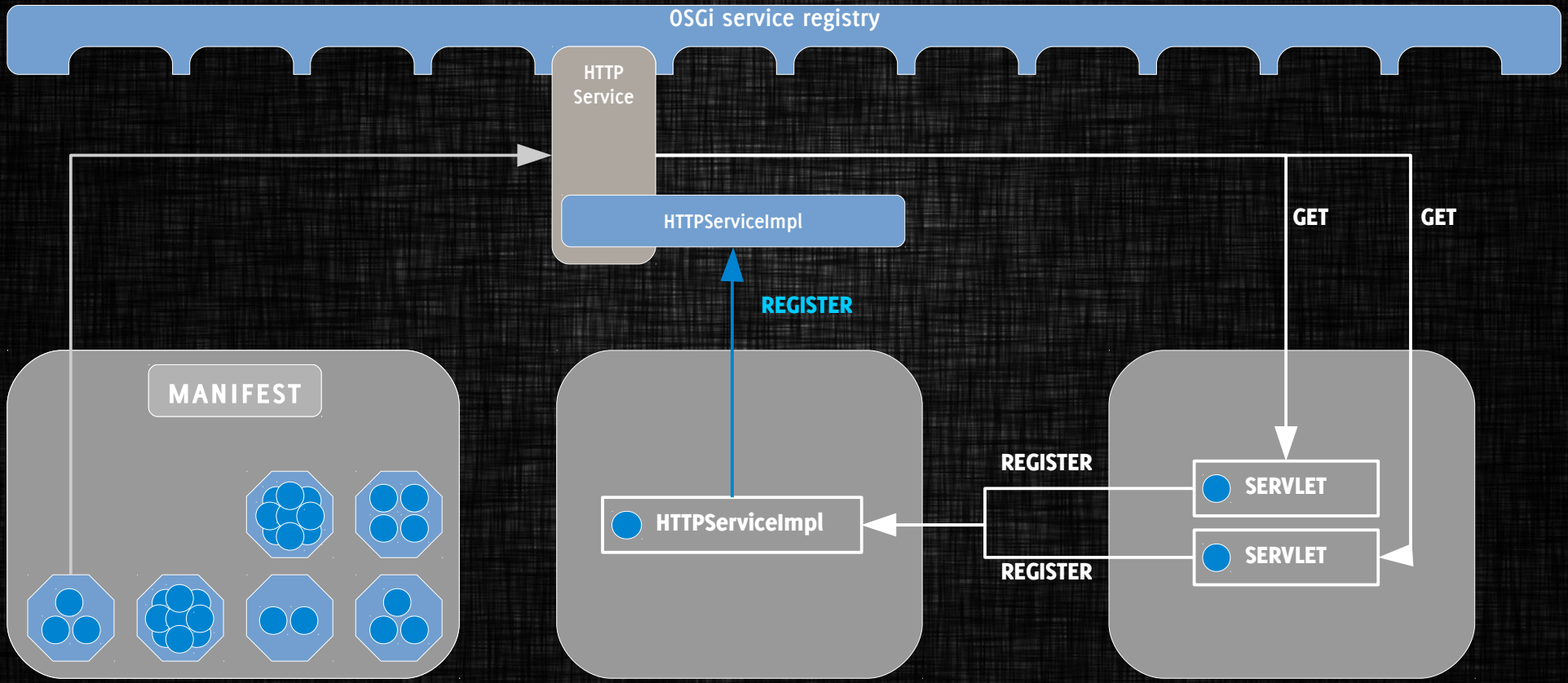
OSGi services



OSGi services

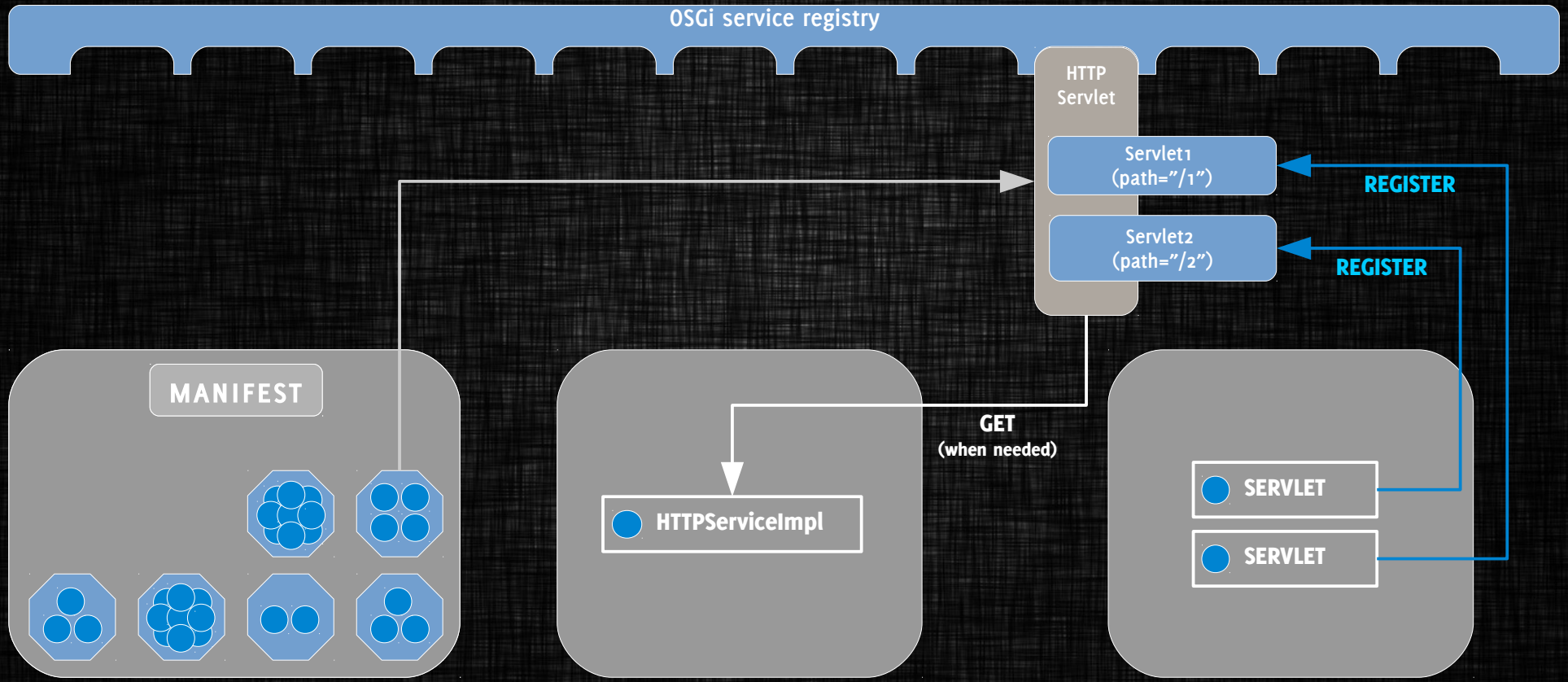


OSGi services

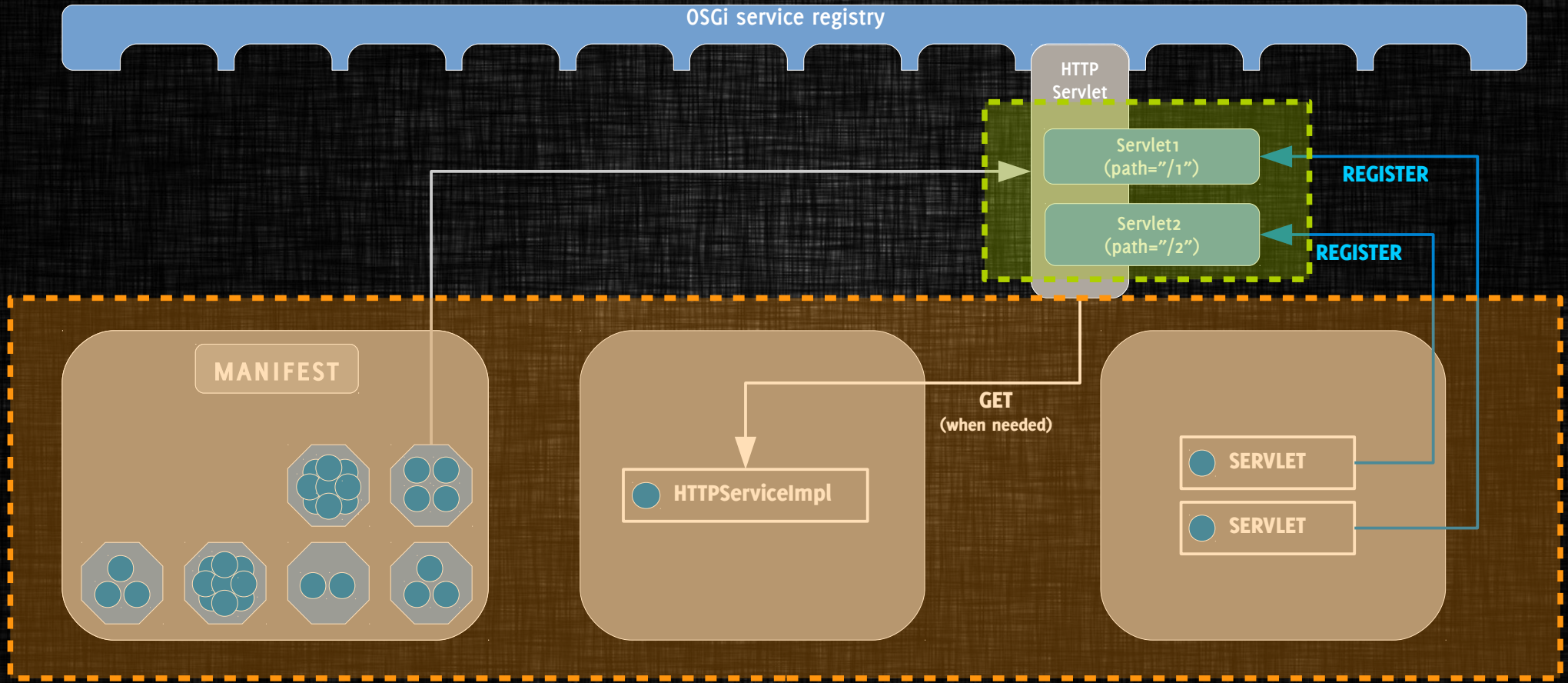


OSGi services

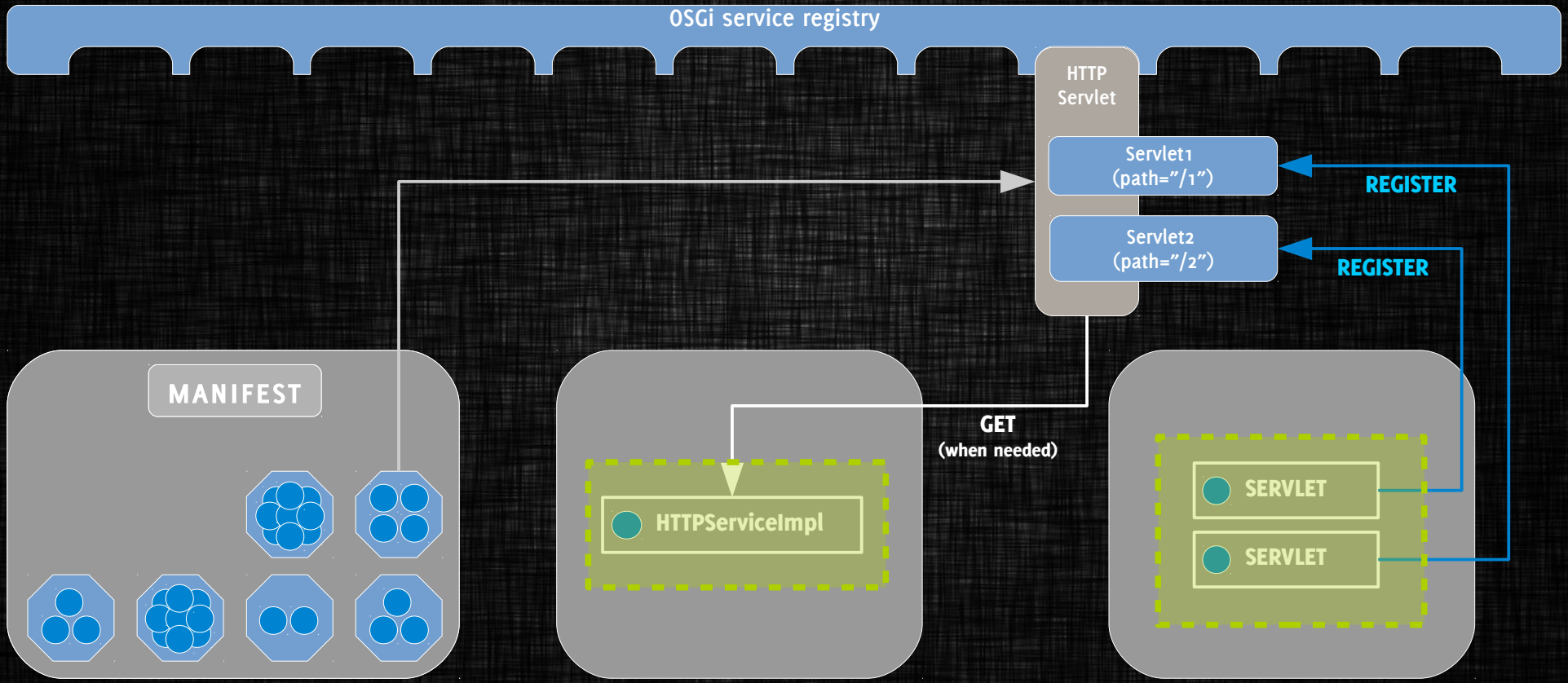
applying White Board Pattern



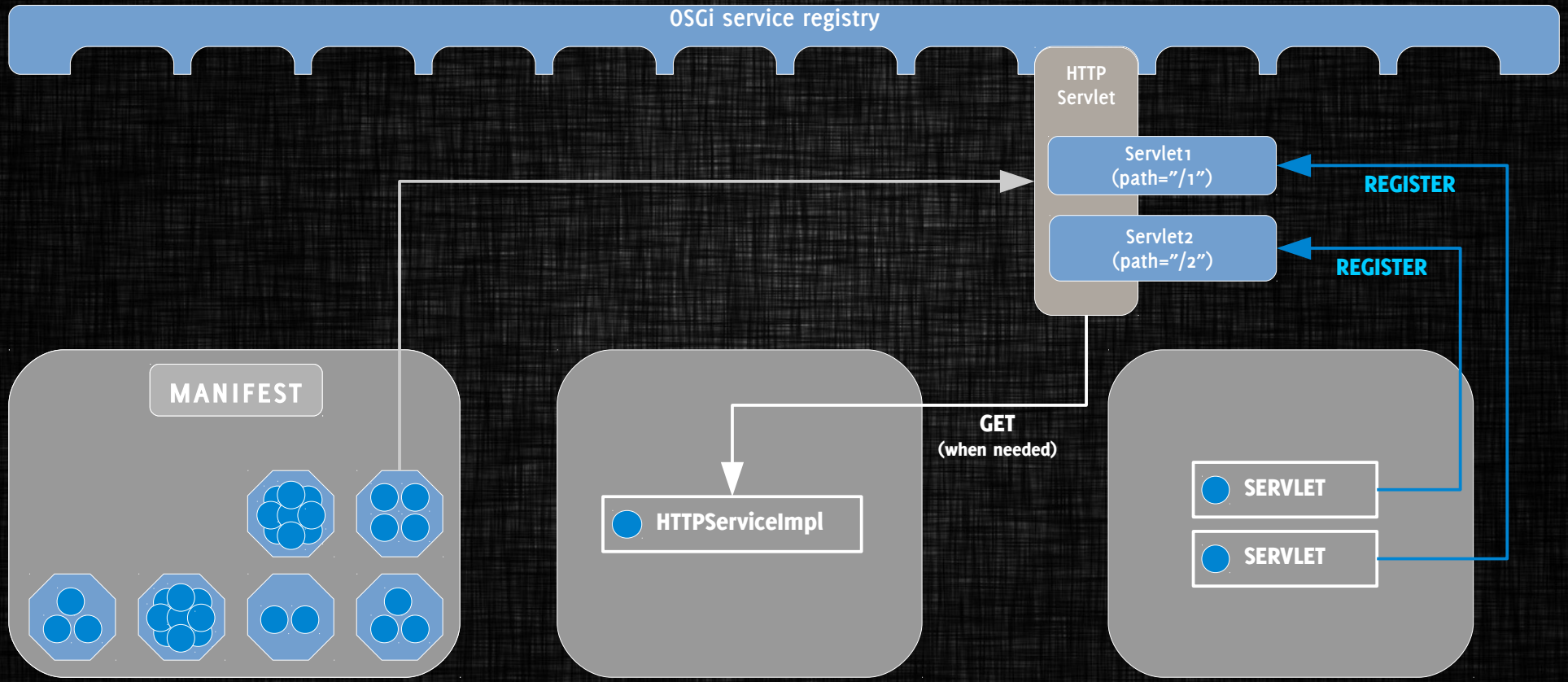
Componentization via Services



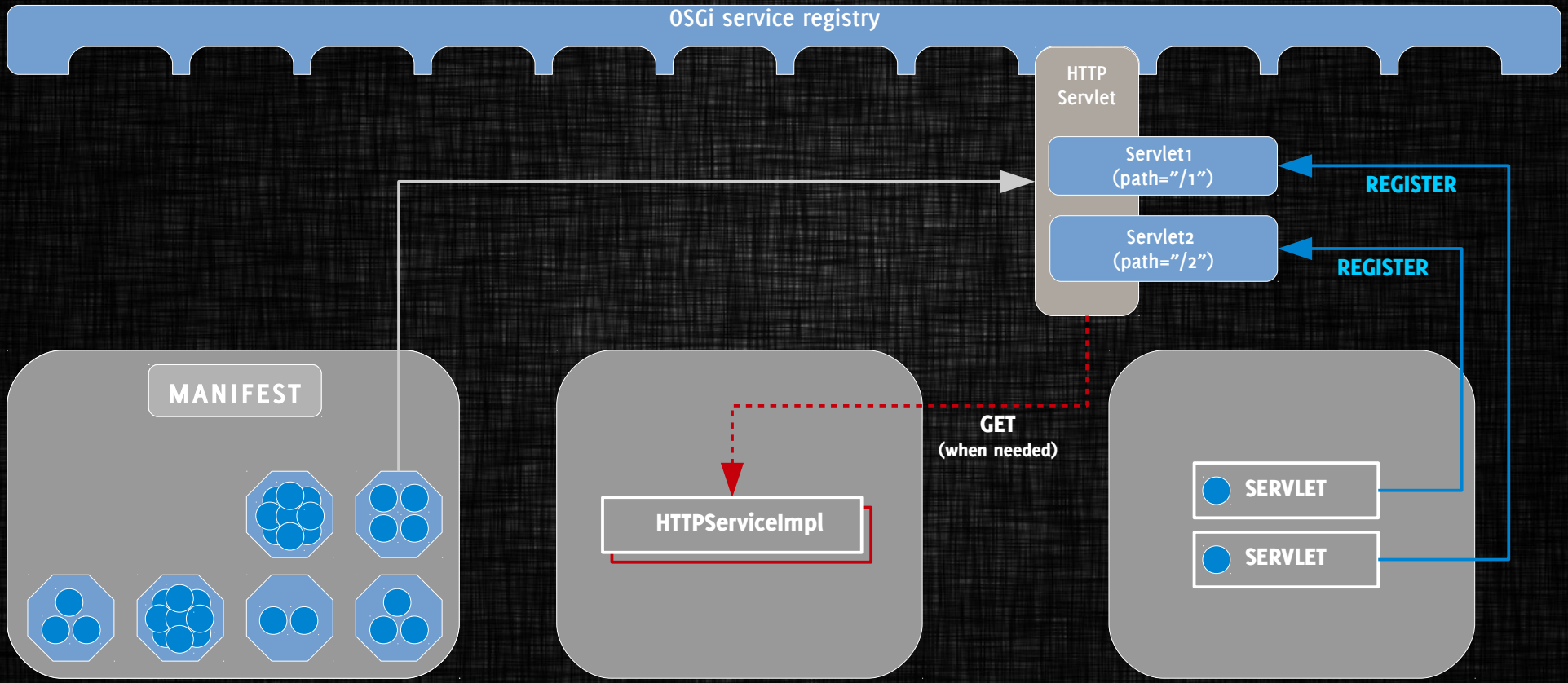
Smart endpoints and ~~dumb~~ no pipes



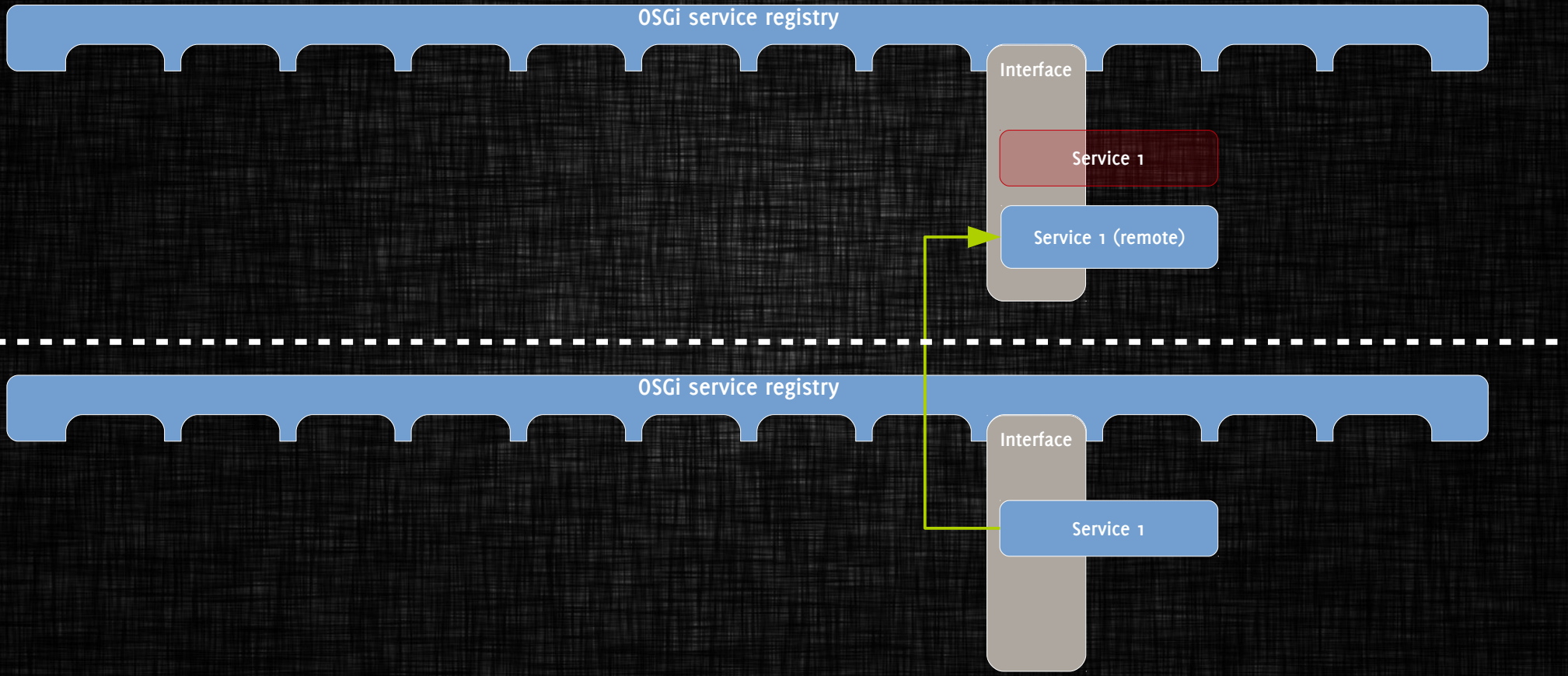
Decentralized ~~Data Management~~ Everything



Design for failure



Remote services



DEMO

milen.dyankov@liferay.com

http://www.liferay.com/web/milen.dyankov/

@milendyankov
@LiferayPL

http://www.liferay.com

@Liferay

http://www.facebook.com/Liferay