WSO2CON 2024

Cloud-Native Middleware: Domain Driven Design, Cell Based Architecture, Service Mesh, and more



Lakmal Warusawithana
Senior Director Cloud Architecture
WSO2

Software Design and Modularity

Modularity

- Modularity, a fundamental principle in software design, plays a crucial role in making software manageable and maintainable.

- It involves breaking down a system into smaller, manageable, and interchangeable components or modules. Each module is designed to execute one specific aspect of the desired functionality and interacts with other modules through well-defined interfaces.

Domain-driven design (DDD)

Domain-driven design (DDD)

- Domain-Driven Design (DDD) is a methodology for software design that involves creating detailed models to capture the business requirements of a specific domain. These models form the conceptual basis for software development.

- DDD helps us build a microservice architecture by splitting a big system into smaller, independent parts. We figure out what each part does and how they connect to each other.

Domain-Driven Design for microservices

- **Strategic phase:** identify the Bounded Contexts and map them out in a context map

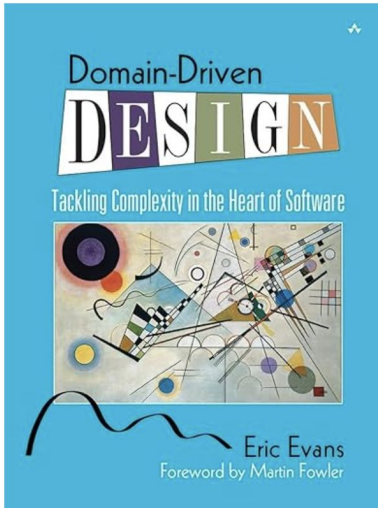


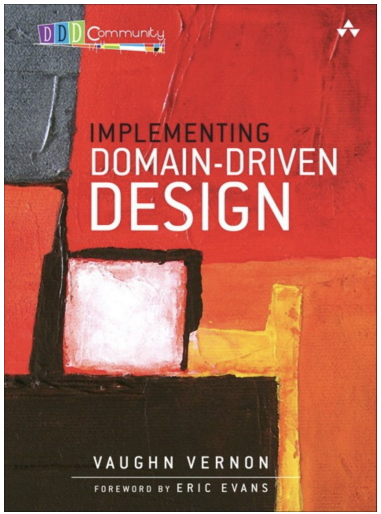
- **Tactical phase:** model each Bounded Context according to the business rules of the subdomain.

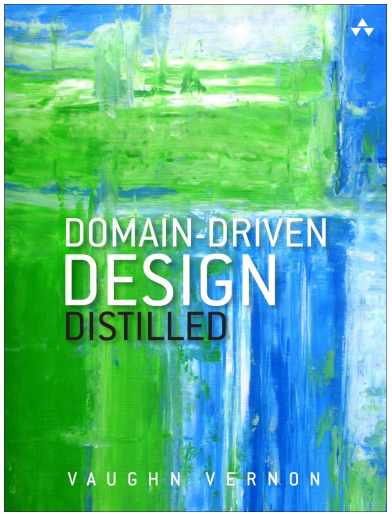
- DDD is iterative and adaptive.

Learn more about Domain-Driven Design

- Domain-Driven Design: Tackling Complexity in the Heart of Software by Eric Evans
- Implementing Domain-Driven Design by Vaughn Vernon
- Domain-Driven Design Distilled by Vaughn Vernon







From Domain-Driven Design to Cloud-Native Deployment

Key Challenges in Transitioning from Domain-Driven Design to Cloud-Native Deployment

Key Challenges in Transitioning from Domain-Driven Design to Cloud-Native Deployment

- Map bounded contexts into the deployment concepts

Key Challenges in Transitioning from Domain-Driven Design to Cloud-Native Deployment

- Map bounded contexts into the deployment concepts
- **Autonomous team operations**

Key Challenges in Transitioning from Domain-Driven Design to Cloud-Native Deployment

- Map bounded contexts into the deployment concepts
- Autonomous team operations
- **Dependency management**

Key Challenges in Transitioning from Domain-Driven Design to Cloud-Native Deployment

- Map bounded contexts into the deployment concepts
- Autonomous team operations
- Dependency management
- **Service discovery and governance**

Key Challenges in Transitioning from Domain-Driven Design to Cloud-Native Deployment

- Map bounded contexts into the deployment concepts
- Autonomous team operations
- Dependency management
- Service discovery and governance
- **Complexity in scaling and resiliency**

Key Challenges in Transitioning from Domain-Driven Design to Cloud-Native Deployment

- Map bounded contexts into the deployment concepts
- Autonomous team operations
- Dependency management
- Service discovery and governance
- Complexity in scaling and resiliency
- **Infrastructure automation and frequence release management**

Key Challenges in Transitioning from Domain-Driven Design to Cloud-Native Deployment

- Map bounded contexts into the deployment concepts
- Autonomous team operations
- Dependency management
- Service discovery and governance
- Complexity in scaling and resiliency
- Infrastructure automation and frequence release management
- **Security implications**

Key Challenges in Transitioning from Domain-Driven Design to Cloud-Native Deployment

- Map bounded contexts into the deployment concepts
- Autonomous team operations
- Dependency management
- Service discovery and governance
- Complexity in scaling and resiliency
- Infrastructure automation and frequence release management
- Security implications
- **Observability and monitoring**

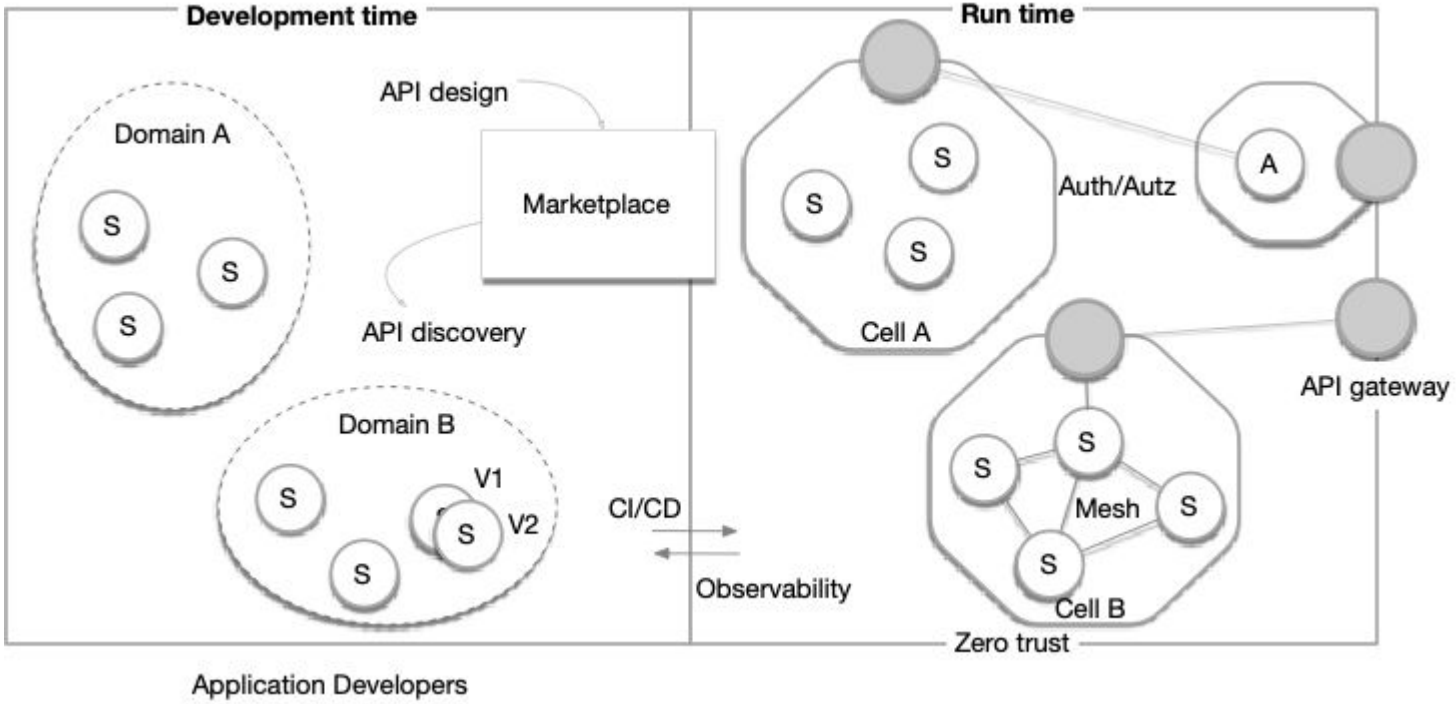
Key Challenges in Transitioning from Domain-Driven Design to Cloud-Native Deployment

- Map bounded contexts into the deployment concepts
- Autonomous team operations
- Dependency management
- Service discovery and governance
- Complexity in scaling and resiliency
- Infrastructure automation and frequence release management
- Security implications
- Observability and monitoring
- **Infrastructure/cloud cost**

By combining the right architecture with cloud-native middleware, these challenges can be effectively overcome.

Reference Architecture

Reference Architecture



https://github.com/wso2/reference-architecture/blob/master/platformless.md

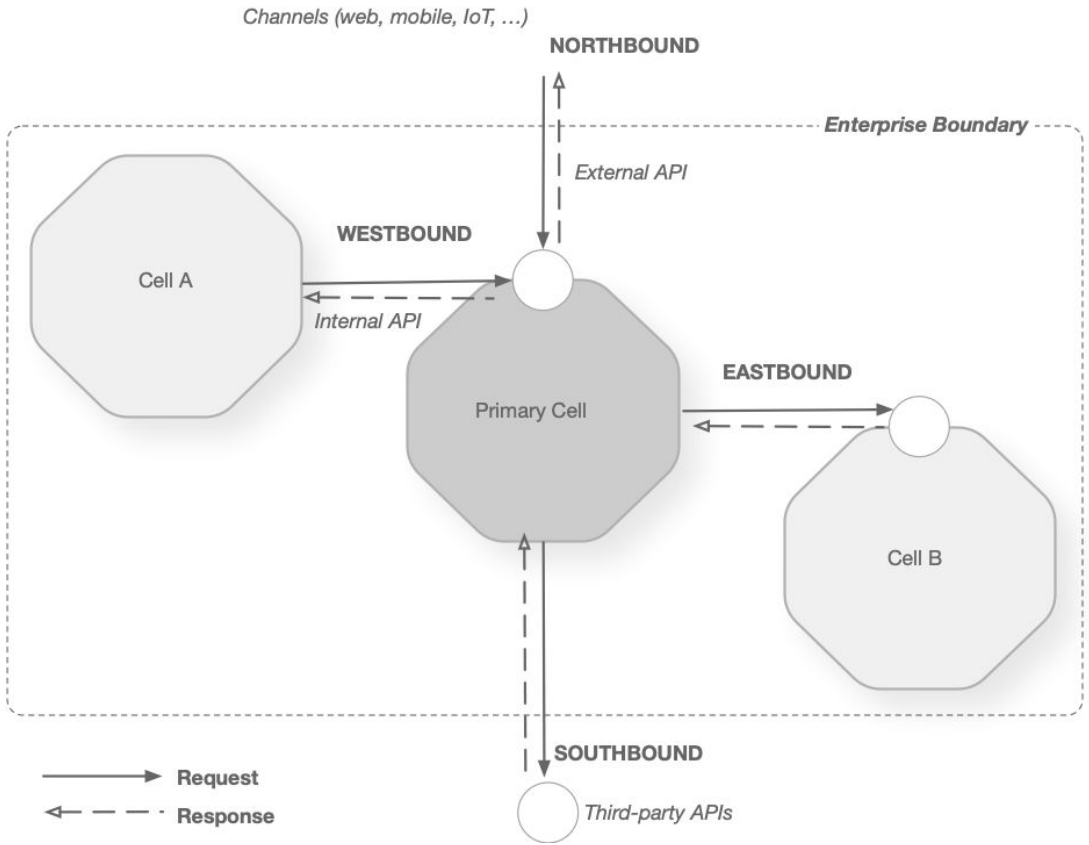
Cell-based architecture (CBA)

Cell-based architecture (CBA)

- A Decentralized Reference Architecture for Cloud-native Applications *by Asanka Abeysinghe, CTO @WSO2 and Paul Fremantle, founder CTO @WSO2*
- Cell-based Architecture (CBA) and Domain-Driven Design (DDD) align well together.
- CBA is an approach for modularizing a group of related capabilities from (part of) a domain into a network cell and managing access to them through well-defined gateways.

https://github.com/wso2/reference-architecture/blob/master/reference-architecture-cell-based.md

Cell-based architecture (CBA)

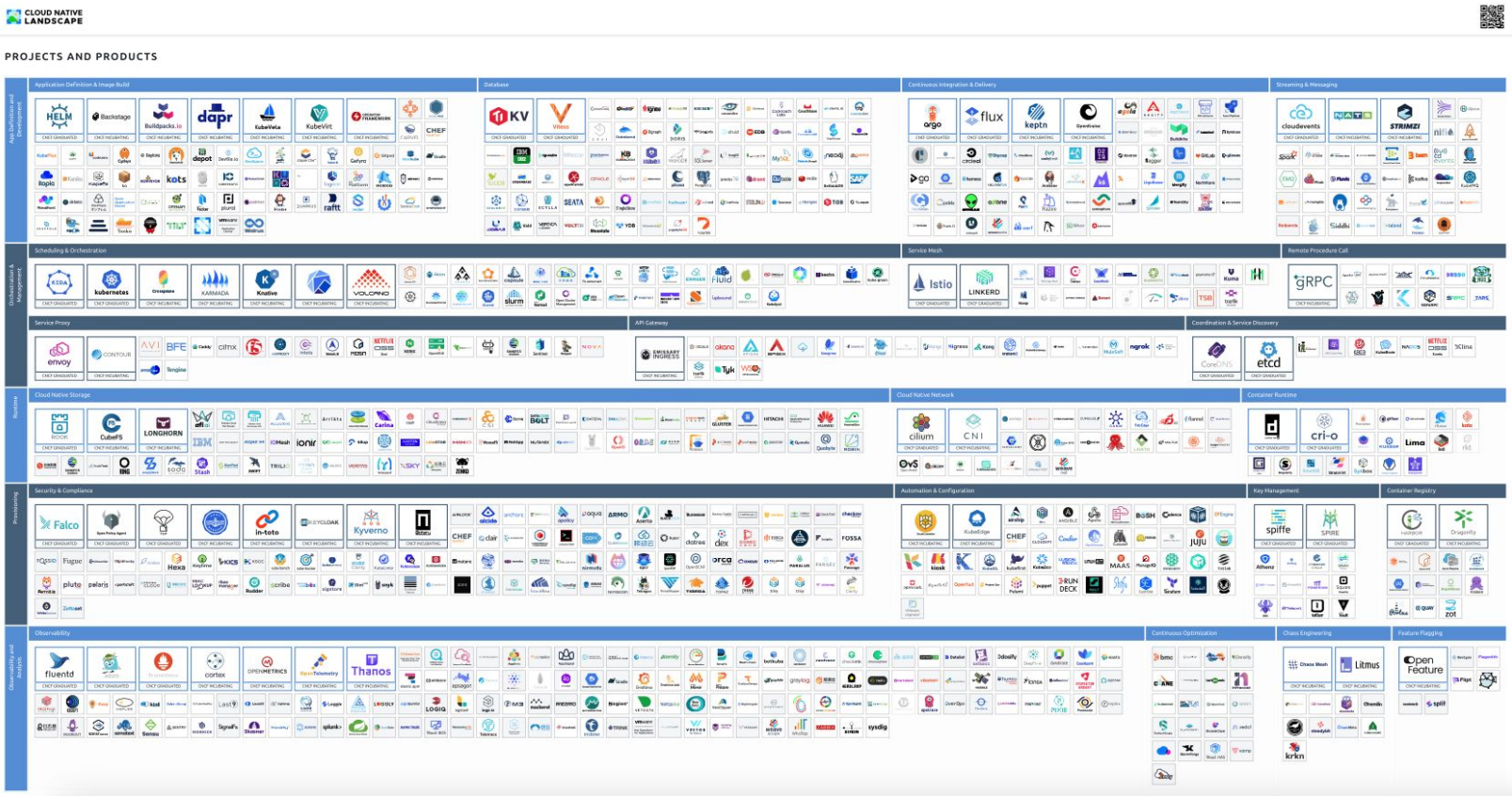


Cloud-Native Middleware

Cloud-Native Middleware

- Cloud-native middleware is a software or set of software tools that provide common services and capabilities to applications beyond those offered by the underlying cloud infrastructure.
- It is designed to support scalability, resilience, maintainability, and observability in cloud-native environments.
- This middleware integrates seamlessly with cloud services and caters to the dynamic nature of cloud-native applications, enabling them to communicate efficiently, scale, and adapt to changing demands without compromising performance or security.
- Additionally, it includes features for monitoring and observing application behavior, which is crucial for troubleshooting and optimizing performance.

Cloud-Native Landscape

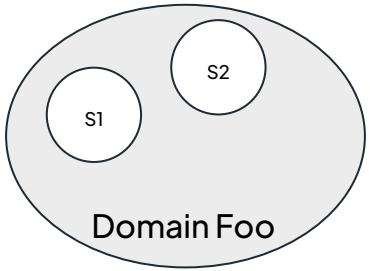


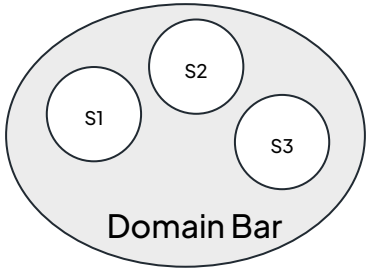
Reference Implementation (Choreo)

Cloud-Native Middleware used in Choreo Implementation

- Github
- Github Runners
- Docker
- Trivy Scanner
- Buildpacks
- Kubernetes
- Key Vault
- API Gateway
- eBPF
- Cilium Service Mesh
- Cilium CNI
- Wireguard
- Hubble
- Envoy
- Prometheus
- Opensearch
- FluentBit
- Keda

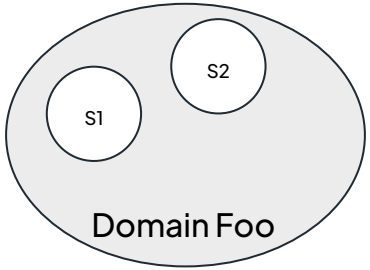
DDD and Dev time

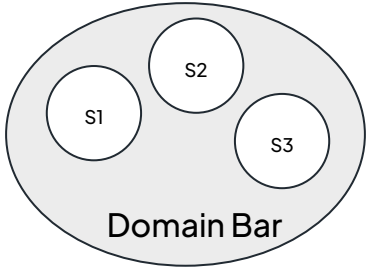




Deployment and Runtime

DDD and Dev time





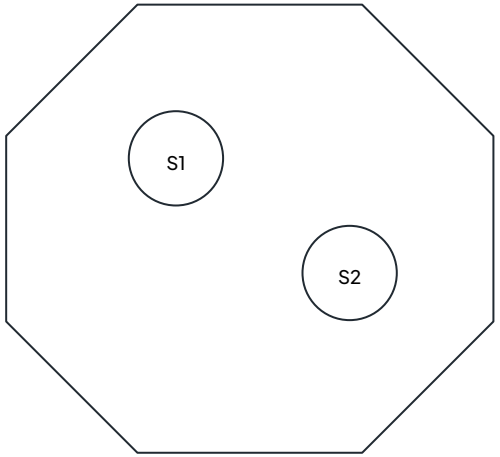
GitHub Docker Registry trivy

docker Buildpacks GitHub Actions

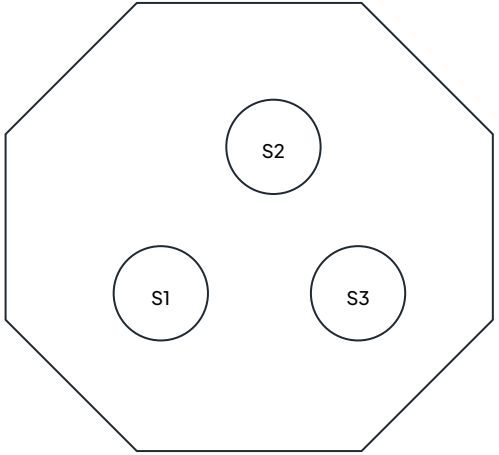
Key Vault

Deployment
Services
Namespaces
HPA
ConfigMaps

Deployment and Runtime



Project Foo



Project Bar

Dev time

Foo
- S1 need to expose to public
- S1->S2

Bar
- S2 need to expose to public
- S2->S3

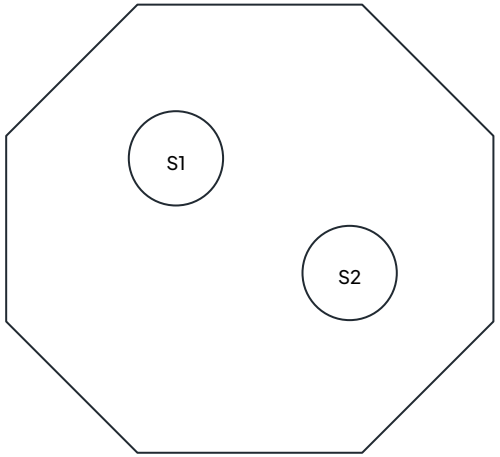
Cross Domain
- Foo/S2->Bar/S1
- Service Discovery

Security
- Authentication and Authorization
- All network call need to be encrypted
- Allow only defined network communication and deny all others.

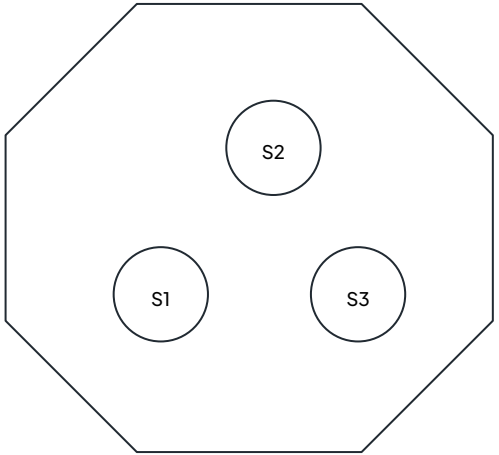
GitHub Docker Registry trivy
docker Buildpacks GitHub Actions

Key Vault
Deployment
Services
Namespaces
HPA
ConfigMaps

Deployment and Runtime



Project Foo



Project Bar

Dev time

Foo
- **S1 need to expose to public**
- **S1->S2**

Bar
- **S2 need to expose to public**
- **S2->S3**

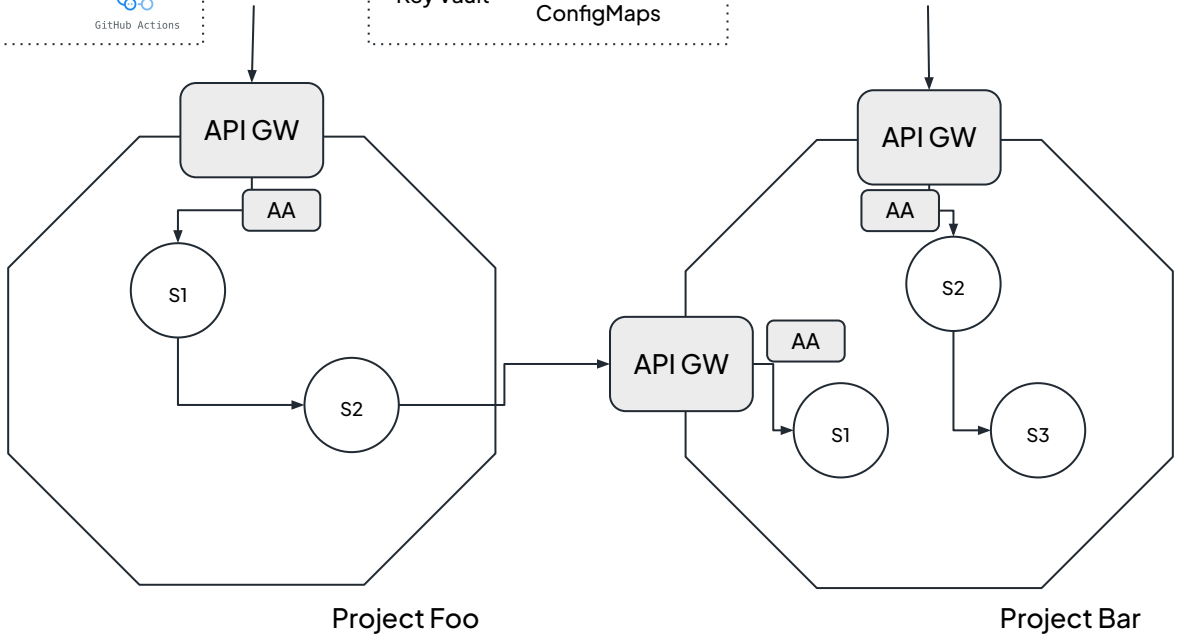
Cross Domain
- **Foo/S2->Bar/S1**
- **Service Discovery**

Security
- **Authentication and Authorization**
- All network call need to be encrypted
- Allow only defined network communication and deny all others.





Deployment and Runtime





Dev time

Foo
- ~~S1 need to expose to public~~
- ~~S1->S2~~

Bar
- ~~S2 need to expose to public~~
- ~~S2->S3~~

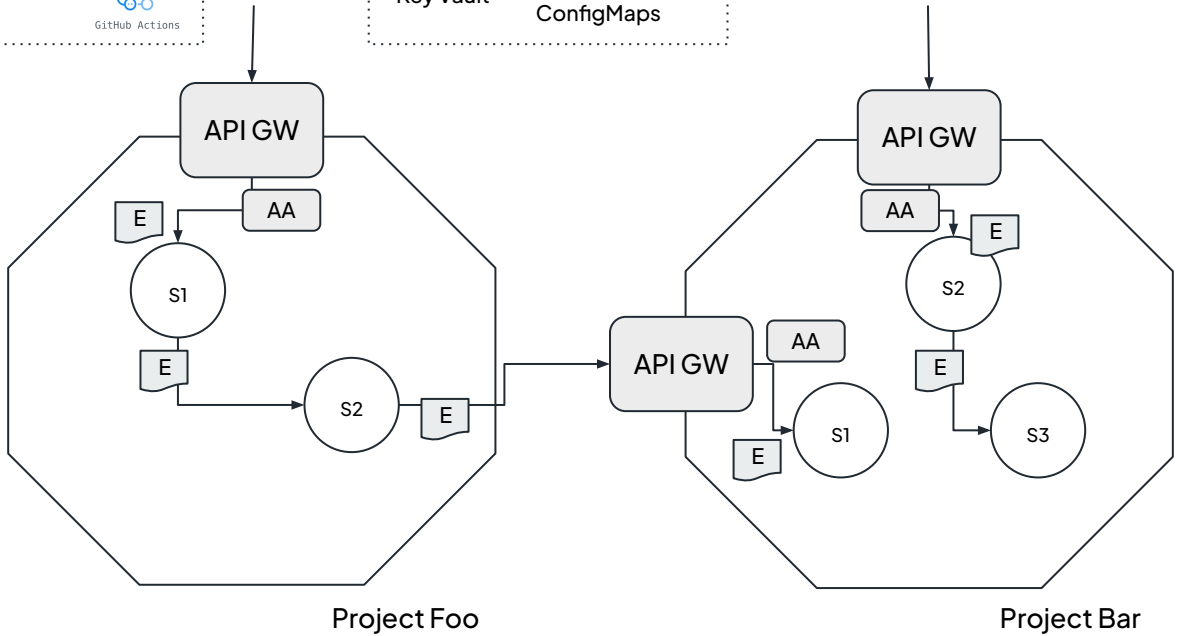
Cross Domain
- ~~Foo/S2->Bar/S1~~
- ~~Service Discovery~~

Security
- ~~Authentication and Authorization~~
- **All network call need to be encrypted**
- Allow only defined network communication and deny all others.

GitHub Docker Registry trivy
docker Buildpacks GitHub Actions

Key Vault
Deployment
Services
Namespaces
HPA
ConfigMaps

Deployment and Runtime



Project Foo

Project Bar

API GW WSO2 envoy
AA Authn & Authz

E Encrypted cilium
WireGuard eBPF

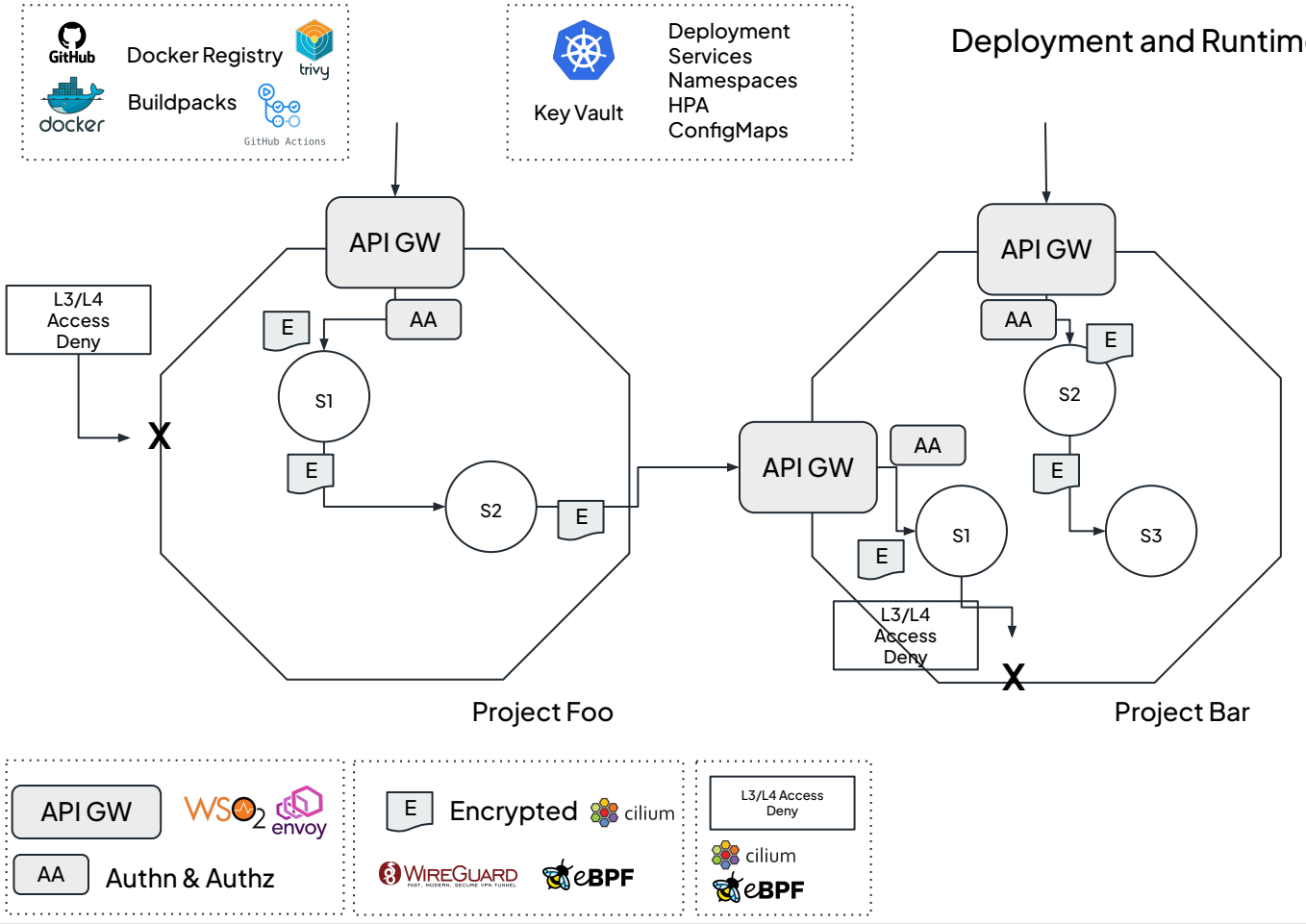
Dev time

Foo
- ~~S1 need to expose to public~~
- ~~S1->S2~~

Bar
- ~~S2 need to expose to public~~
- ~~S2->S3~~

Cross Domain
- ~~Foo/S2->Bar/S1~~
- ~~Service Discovery~~

Security
- ~~Authentication and Authorization~~
- ~~All network call need to be encrypted~~
- **Allow only defined network communication and deny all others.**



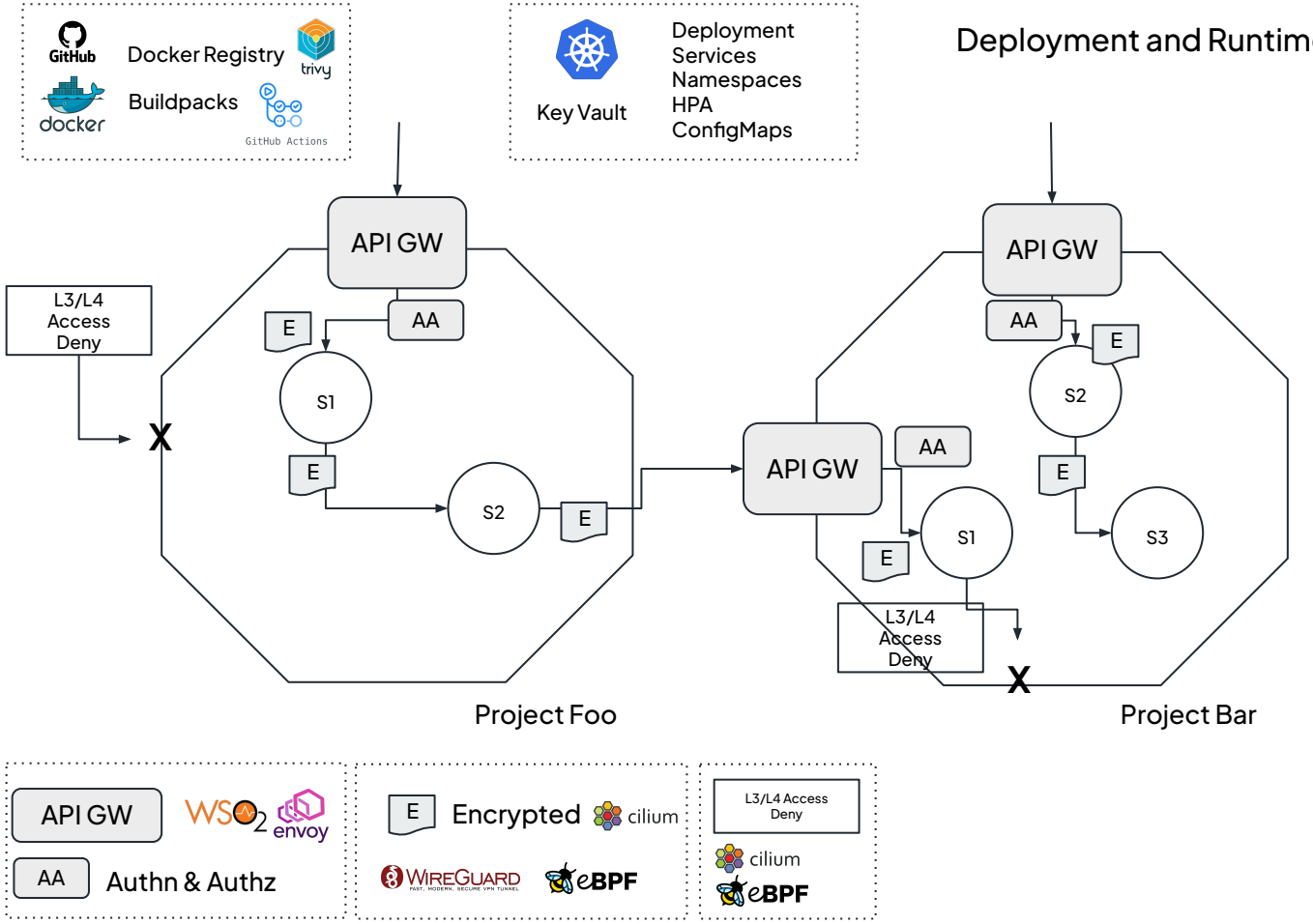
Dev time

Observability
- Logs: access, debugs etc
- System metrics: memory, cpu etc
- Runtime metrics: network traces, request counts, latency, errors etc

Service Mesh
- Resiliency: retry, circuit breaker etc

Serverless
- Scale to zero

Deployment and Runtime



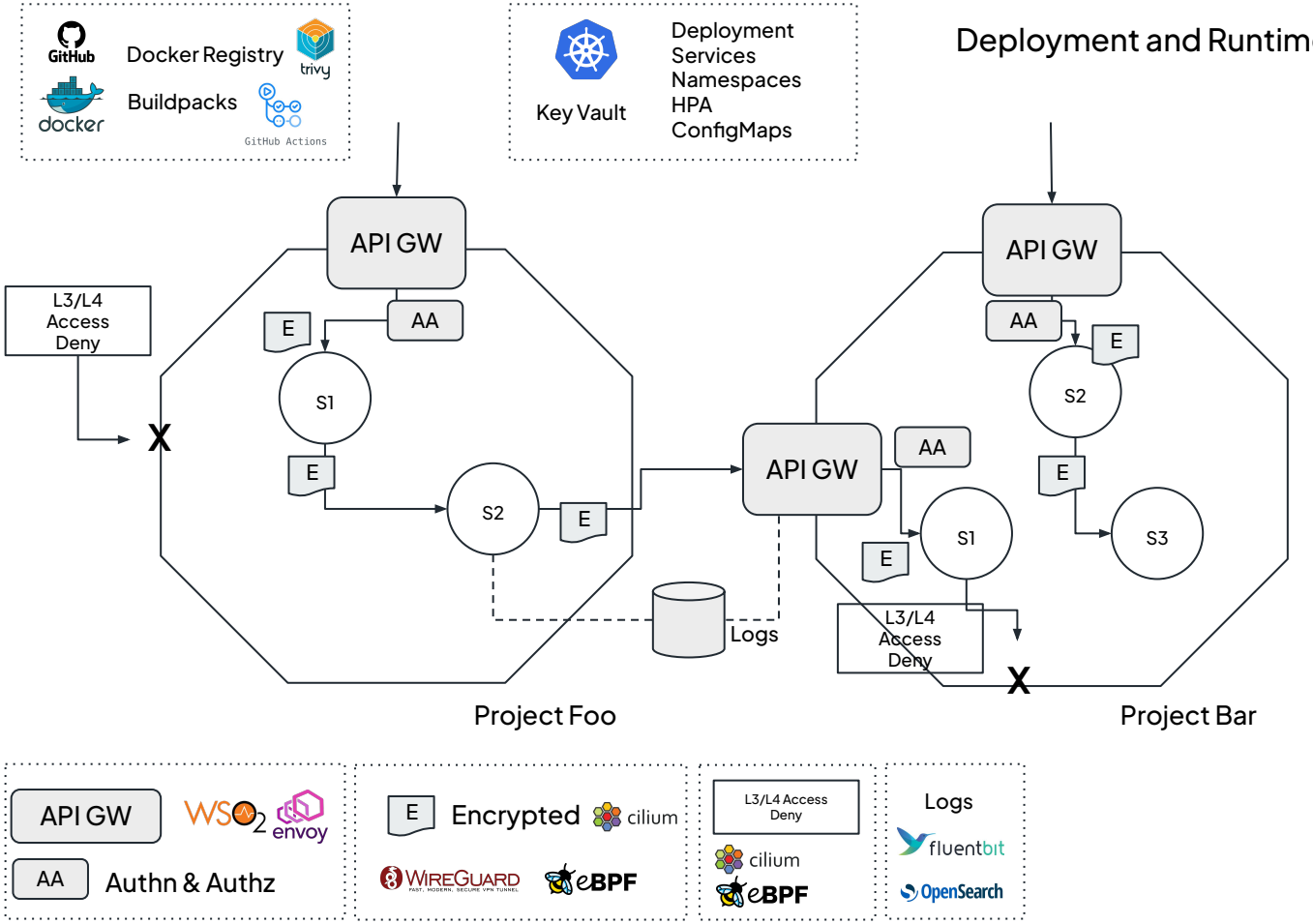
Dev time

Observability
- **Logs: access, debugs etc**
- System metrics: memory, cpu etc
- Runtime metrics: network traces, request counts, latency, errors etc

Service Mesh
- Resiliency: retry, circuit breaker etc

Serverless
- Scale to zero

Deployment and Runtime



Dev time

Observability

- ~~Logs: access, debugs etc~~
- **System metrics: memory, cpu etc**
- **Runtime metrics: network traces, request counts, latency, errors etc**

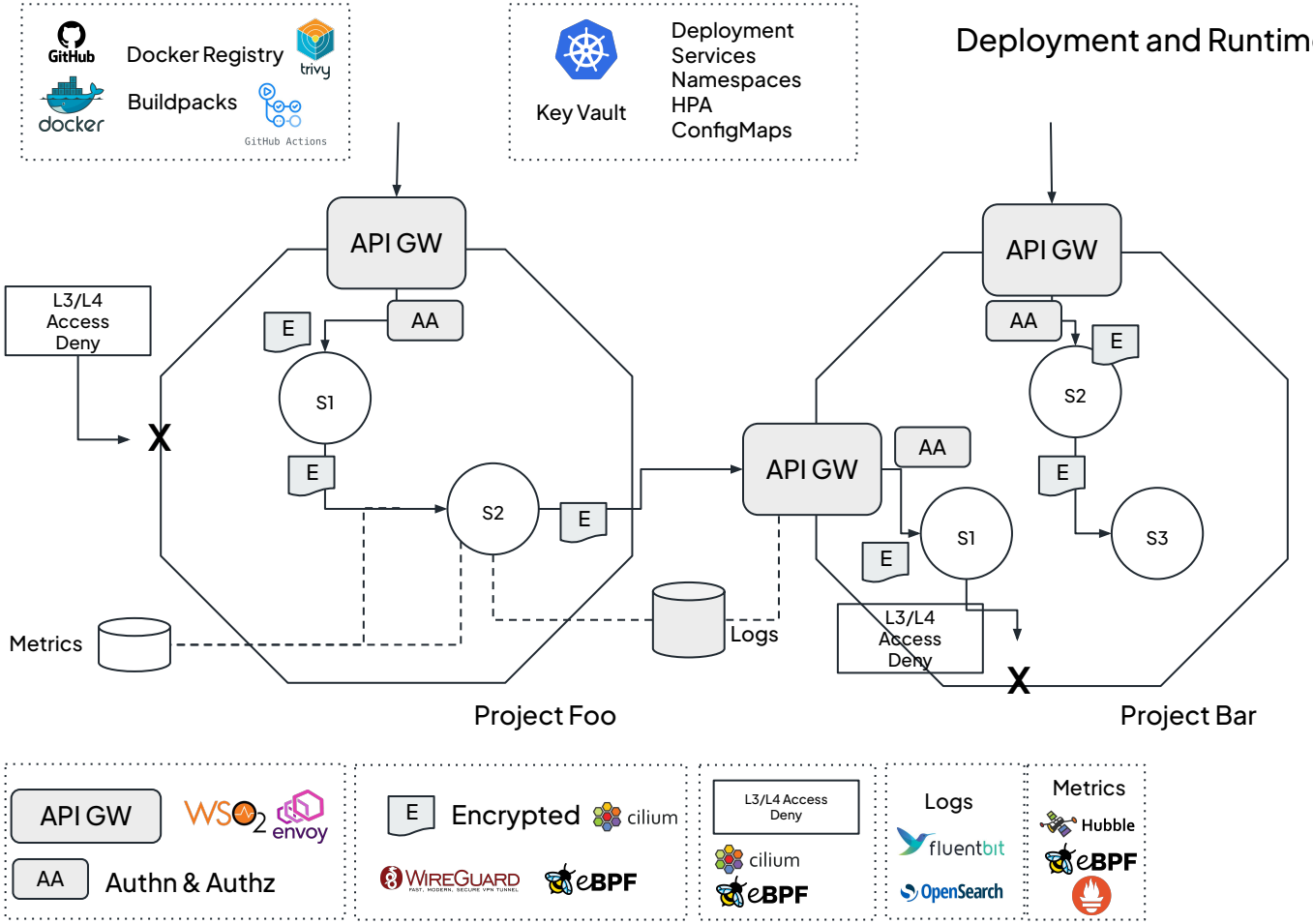
Service Mesh

- Resiliency: retry, circuit breaker etc

Serverless

- Scale to zero

Deployment and Runtime



Dev time

Observability

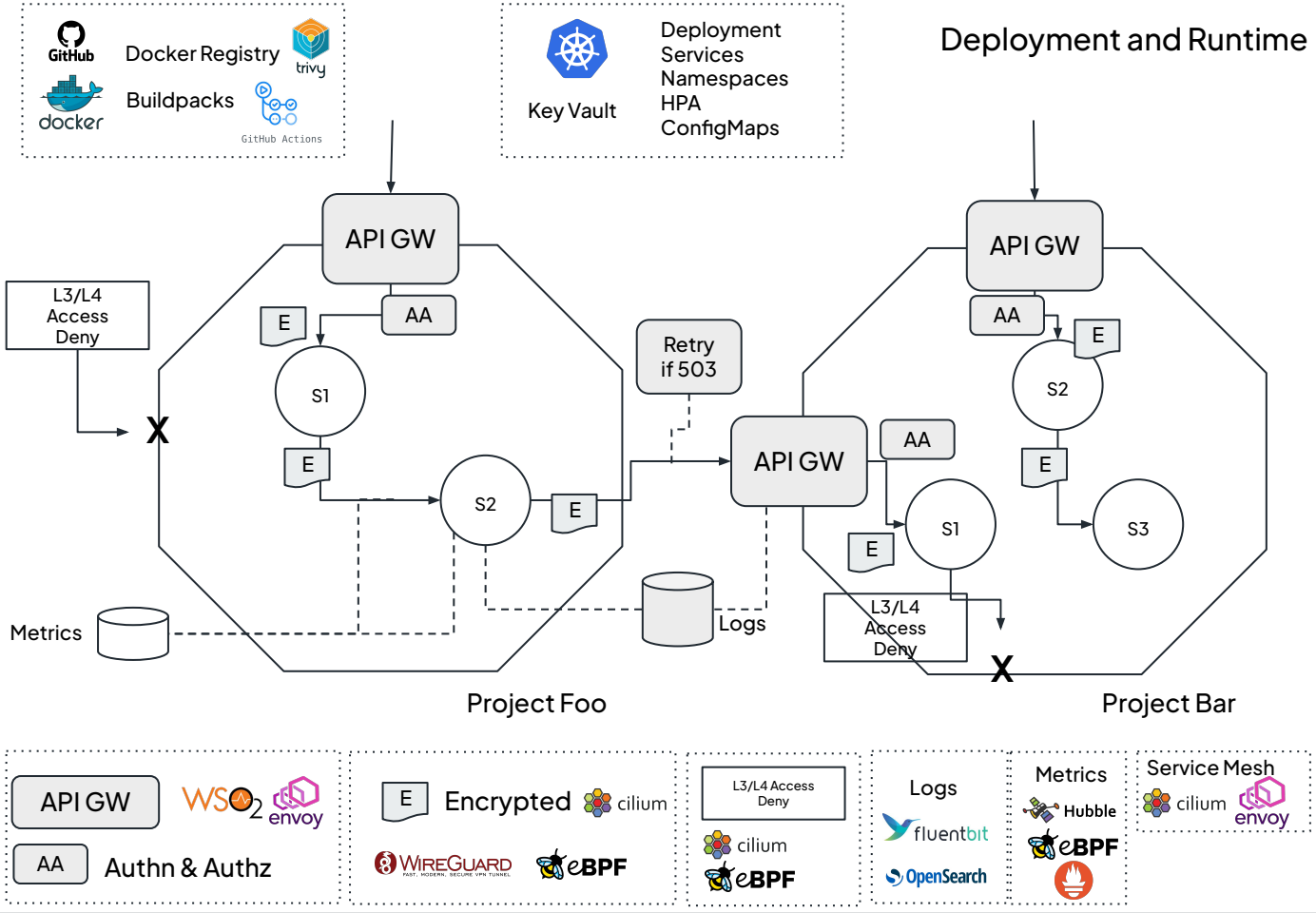
- ~~Logs: access, debugs etc~~
- ~~System metrics: memory, cpu etc~~
- ~~Runtime metrics: network traces, request counts, latency, errors etc~~

Service Mesh

- **Resiliency: retry, circuit breaker etc**

Serverless

- Scale to zero



Dev time

Observability

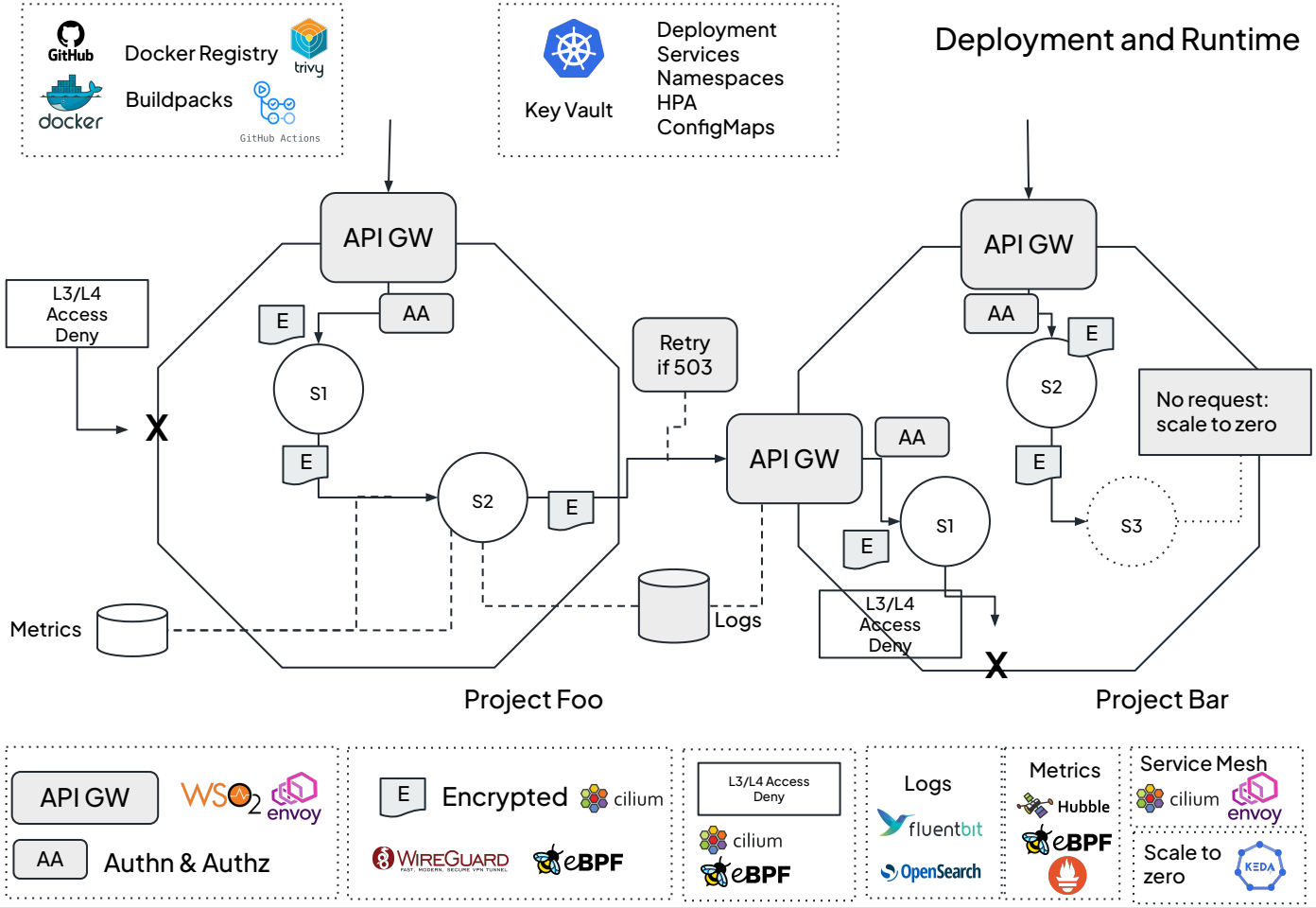
- ~~Logs: access, debugs etc~~
- ~~System metrics: memory, cpu etc~~
- ~~Runtime metrics: network traces, request counts, latency, errors etc~~

Service Mesh

- ~~Resiliency: retry, circuit breaker etc~~

Serverless

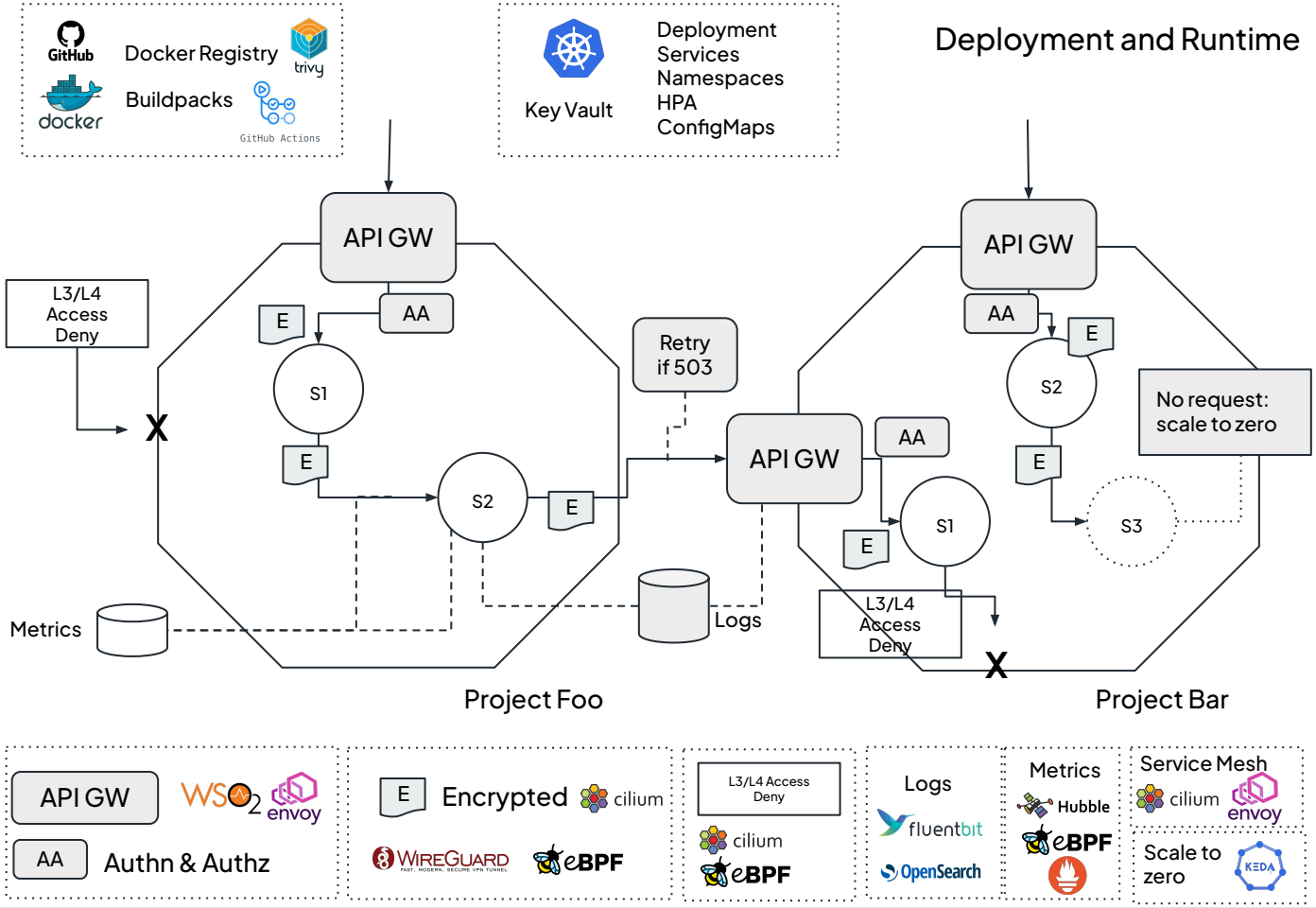
- **Scale to zero**



Zero Trust Principals

- Never Trust, Always Verify
- Verify Explicitly
- Least Privilege Access
- Micro-Segmentation
- Layered Security Controls and MFA
- Assume Breach
- End-to-End Encryption
- Full User and System Visibility

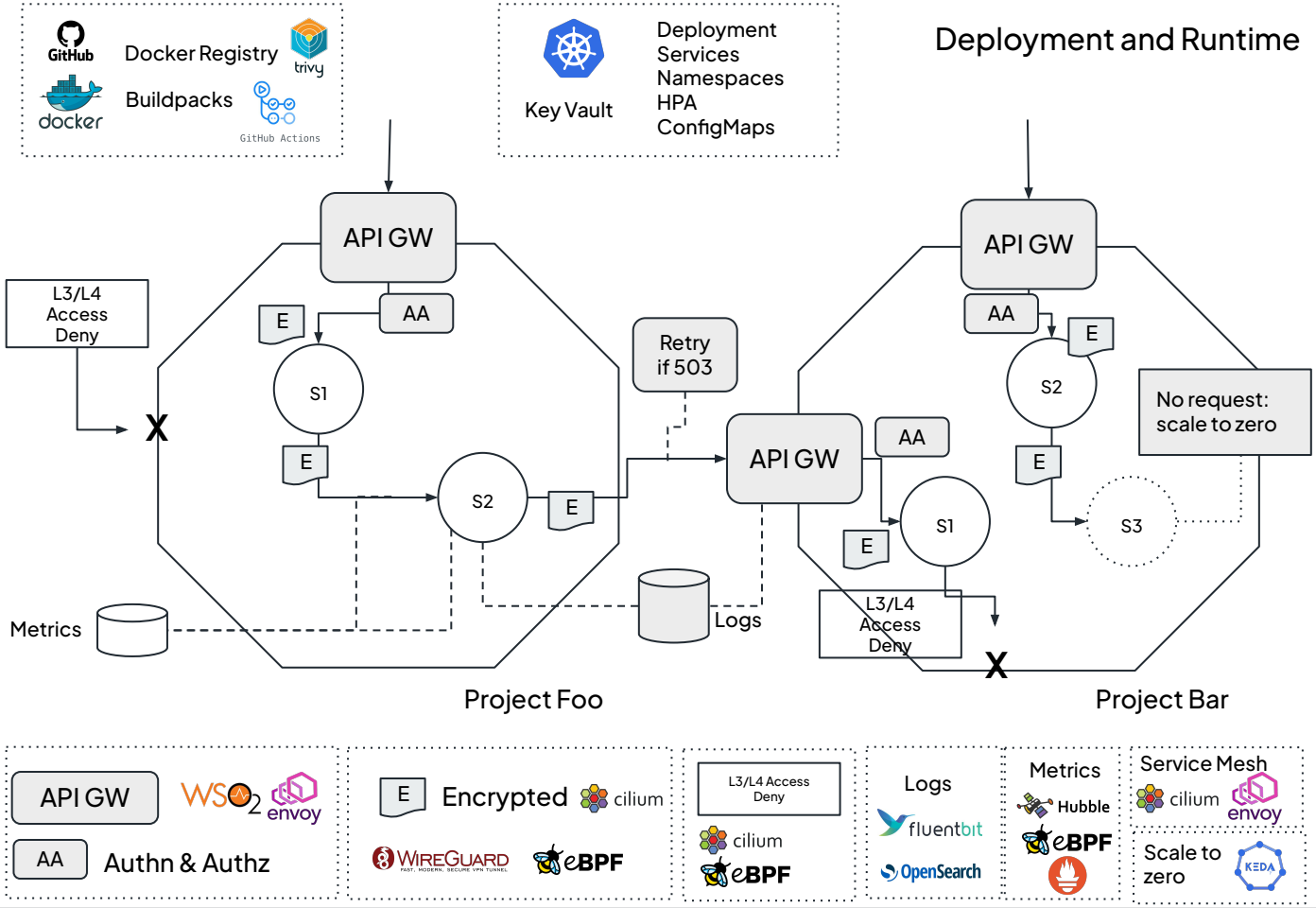
Deployment and Runtime



Revisit Challengers

- Map bounded contexts into the deployment concepts
- Autonomous team operations
- Dependency management
- Service discovery and governance
- Complexity in scaling and resiliency
- Infrastructure automation frequence release management
- Security Implications
- Observability, Monitoring and Maintenance
- Infrastructure/cloud cost

Deployment and Runtime



After building this platform, you'll streamline the development process, enhance productivity, and accelerate time-to-market, all of which are essential for business success.

Platformless

Read more…

- Platformless *by Sanjiva, Paul and Asanka*
- Cell-Based Architecture *by Paul and Asanka*
- Unlocking the Power of Programmable Data Planes in Kubernetes with eBPF *by Lakmal*
- How We Implemented Zero Trust in Choreo *by Lakmal*
- Scaling Smart: Introducing Choreo's Scale-to-Zero for Optimal Resource Utilization *by Lakmal*
- Reference Architecture for a Cloud Native Digital Enterprise *by Lakmal*
- Reference Implementation for a Cloud Native Digital Enterprise *by Lakmal*

Question Time!



Thank You!