Kalix is the PaaS that enables any developer to easily build large-scale, high performance microservices with no operations required.
Kalix Fundamentals

**Self-Managed On-Prem**
- Application
  - Business Logic
  - Frameworks
  - Database
  - Transport
  - Security
- Infrastructure
  - Kubernetes
  - Operating System
  - Virtualization
  - Servers, Storage, Networking

**In the Cloud**
- Application
  - Business Logic
  - Frameworks
  - Database
  - Transport
  - Security
- Infrastructure
  - Kubernetes
  - Operating System
  - Virtualization
  - Servers, Storage, Networking

**Kalix**
- Application
  - Business Logic
  - Frameworks
  - Database
  - Transport
  - Security
- Infrastructure
  - Kubernetes
  - Operating System
  - Virtualization
  - Servers, Storage, Networking

*Self-Managed*  
*Managed by Cloud / Service Provider*
Kalix Fundamentals

Built on Akka

- Performance
- Efficiency
- Reliability
- Real-Time
- Scalability
- Streaming
Kalix Fundamentals

- Serverless Runtime
- Built on Akka
- Managed runtime infrastructure
- Container-based deployments
- CI/CD integrations
- Logging & Monitoring
- Auto-Scaling
- Security
Kalix Fundamentals

Capabilities

- Stateful data handling
- Communication patterns
- Event Sourcing and CQRS patterns

Serverless Runtime

Built on Akka
Kalix Fundamentals

Polyglot language support

- Capabilities
- Serverless Runtime
- Built on Akka

Languages supported: Java, TypeScript, JavaScript
How Do You Build a Service?

1. API description
2. Durable storage model
3. Business logic
How Do You Build a Service?

- API first development model
- Design API endpoints
  - Just configure required communication patterns (HTTP, gRPC, RPC, streaming, eventing)
- Design persistence domain
- Focus on data structure, not database
How Do You Build a Service?

Choose a durable store model

- Key/Value, Event Sourcing, CRDTs
- Simple model change by annotation

```java
//service
service ShoppingCartService {
    option (kalix.codegen) = {
        event_sourced_entity: {...}
    };
    rpc AddItem (AddItemRequest) returns (Cart);
    rpc CheckOut (CheckOutRequest) returns (Cart);
}
```
How Do You Build a Service?

- Focus on business logic – implement function
- Data (state) automatically injected in the function – no DB access
Under the Hood

Execution cluster

Project

State proxy
User code

kubernetes
Under the Hood

Every proxy is an Akka node...
Future of Serverless Computing

• How do you see Serverless Computing in five years?
  – DX for new wave of cloud and edge native apps and use-cases

• Propose a technical challenge to solve in this field in the next years
  – new architecture for the edge with more fine-grained data replication, closer to the end user, physical co-location of data, processing, and user
  – compose cloud and edge in a single coherent whole

• What question would you want to ask another participant?
  – How can we tackle security compliances in serverless environments?