
AWS
re:Invent
NOV. 28 – DEC. 2, 2022 | LAS VEGAS, NV

PRT301

SPONSORED BY CAPGEMINI

Smart serverless architecture for smart manufacturing

Jens Busch (he/him)

DPP Solution Architect
Audi

Markus Großmann (he/him)

Solution Architect Director
XL2 by Audi & Capgemini

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Texts (excl. quotes), presentation concept and selected logos/images/icons (excl. AWS icons): Copyright © 2022, Audi AG / XL2 GmbH. All rights reserved.

About us



Audi

Jens Busch

IT Solutions Architect for digital production platform

More than two decades in automotive IT

Deep knowledge of shop-floor-based software systems



XL2 by Audi & Capgemini

Markus Großmann

Cloud team chief architect

Located in Heilbronn, Germany

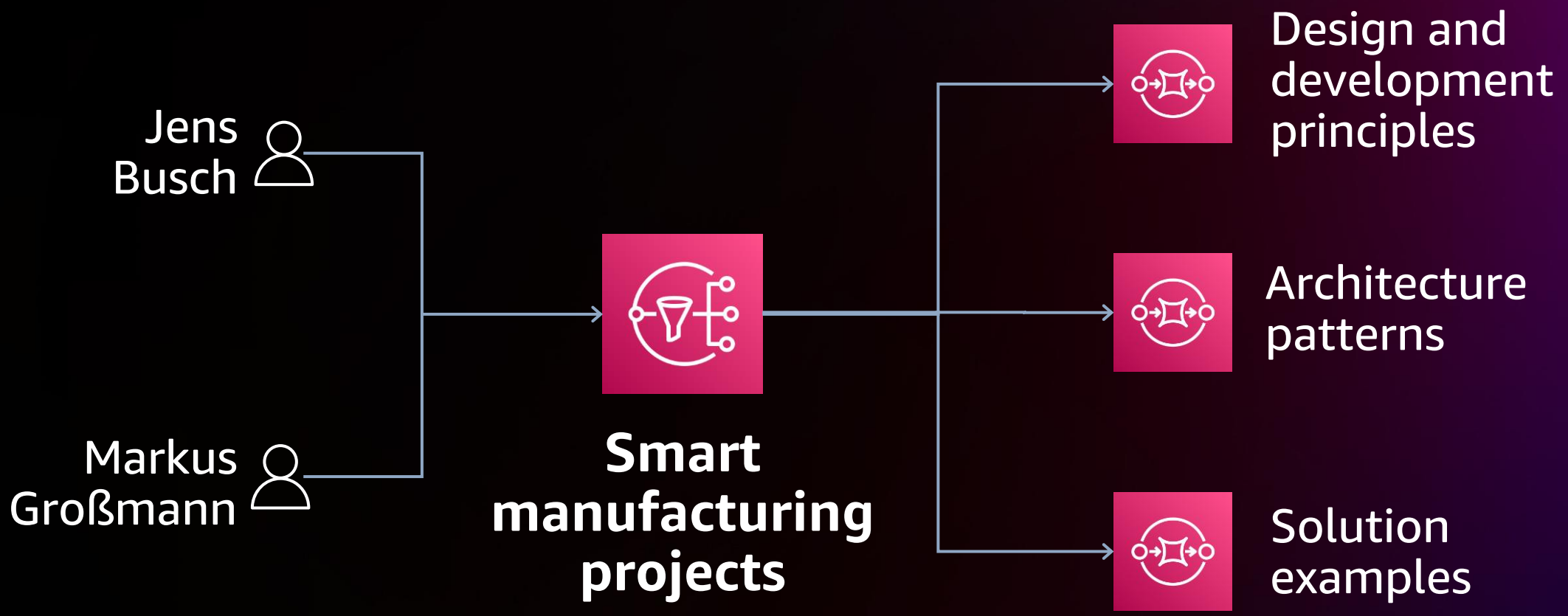
Passionate musician – love being creative and playing with others

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Texts (excl. quotes), presentation concept and selected logos/images/icons (excl. AWS icons): Copyright © 2022, Audi AG / XL2 GmbH. All rights reserved.

What are we talking about today?



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Texts (excl. quotes), presentation concept and selected logos/images/icons (excl. AWS icons): Copyright © 2022, Audi AG / XL2 GmbH. All rights reserved.

Smart manufacturing is a broad topic

WE FOCUS ON PRODUCTION AND LOGISTICS SYSTEMS ON TOP OF THE SHOP-FLOOR LAYER



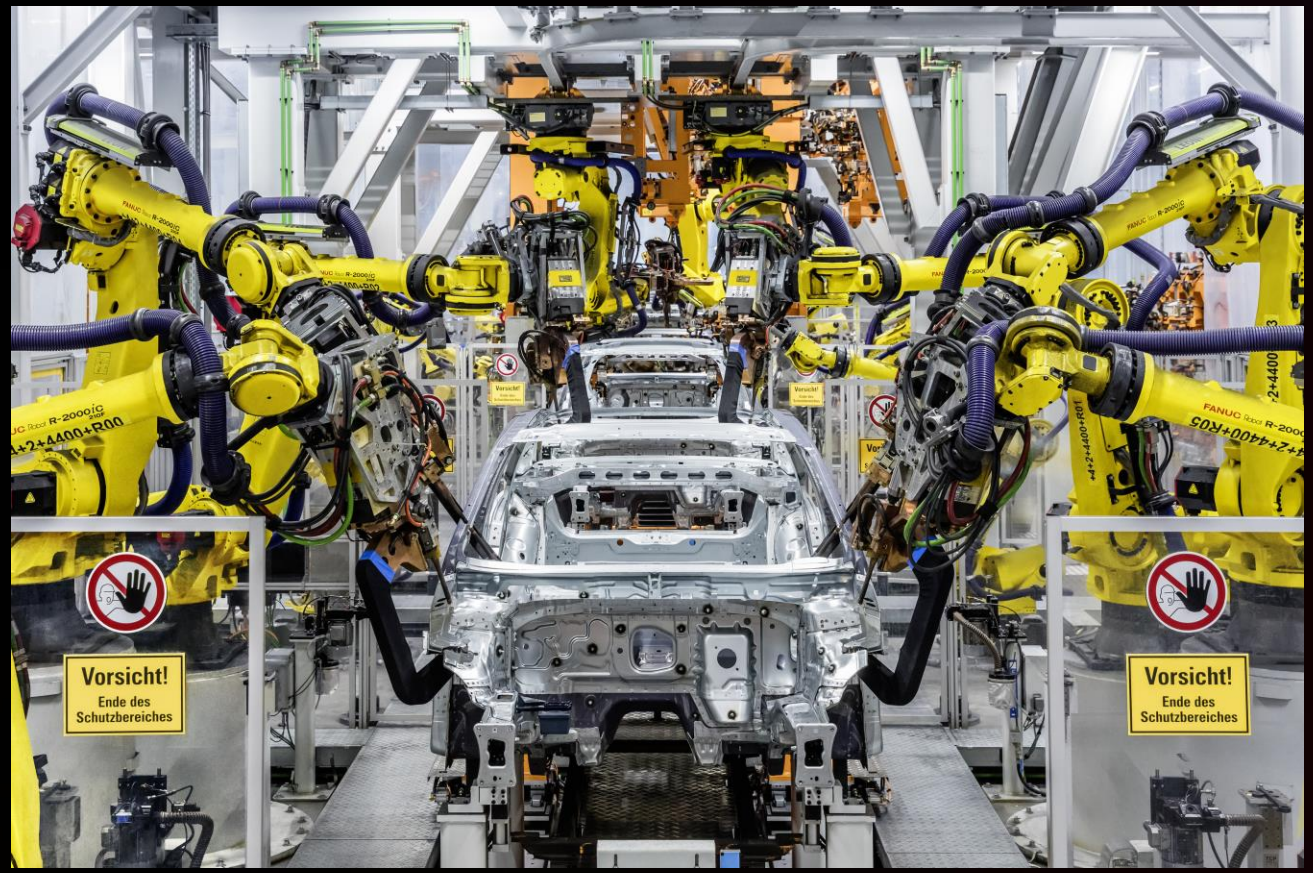




© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Texts (excl. quotes), presentation concept and selected logos/images/icons (excl. AWS icons): Copyright © 2022, Audi AG / XL2 GmbH. All rights reserved.

Real use case: Parameter service



- Transmits car-specific setting parameters to production line stations
- Stores and visualizes assembly process results
- Implemented with AWS IoT Greengrass and serverless services

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Texts (excl. quotes), presentation concept and selected logos/images/icons (excl. AWS icons): Copyright © 2022, Audi AG / XL2 GmbH. All rights reserved.

Topics we will cover

Architectural challenges in smart manufacturing

Design principles and patterns

- Component-oriented design and service stores
- Elastic data models
- Closed feedback loops

Development principles for serverless projects

Summary

Q&A

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Texts (excl. quotes), presentation concept and selected logos/images/icons (excl. AWS icons): Copyright © 2022, Audi AG / XL2 GmbH. All rights reserved.

Architectural challenges in smart manufacturing

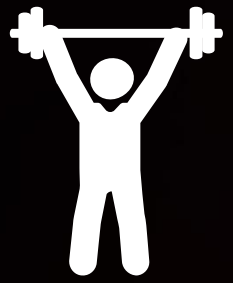
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

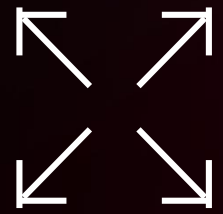
Texts (excl. quotes), presentation concept and selected logos/images/icons (excl. AWS icons): Copyright © 2022, Audi AG / XL2 GmbH. All rights reserved.

Manufacturing systems aren’t boring

TYPICAL CHALLENGES









24/7 availability

Resilient

Scalable

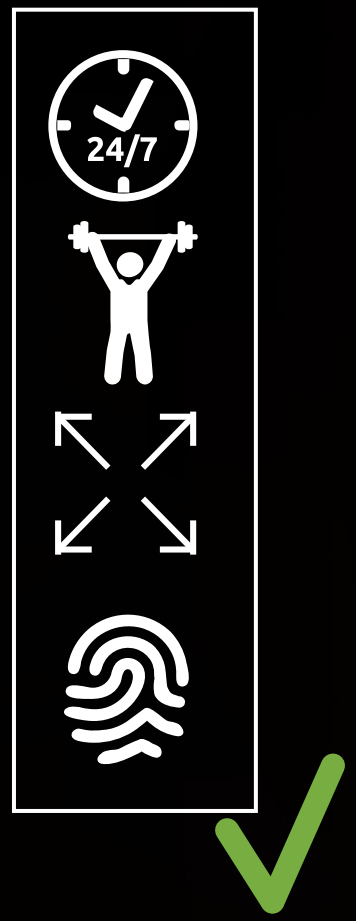
Secure

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Texts (excl. quotes), presentation concept and selected logos/images/icons (excl. AWS icons): Copyright © 2022, Audi AG / XL2 GmbH. All rights reserved.

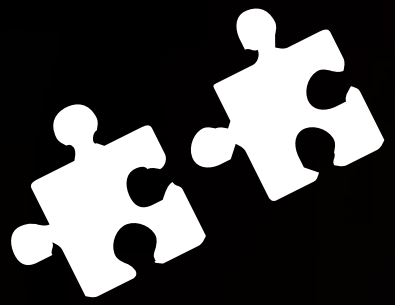
Serverless is generally a good architectural answer



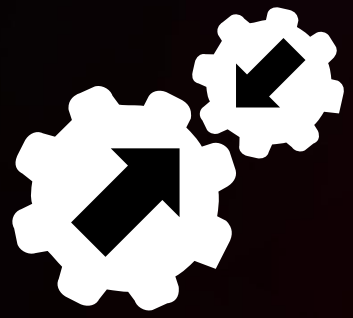
- Highly available and inherently scalable
- Supports typical event-driven workloads well
- Comprehensive package of ready-to-integrate cloud services
- High durability and built-in error handling
- Security and compliance benefit from managed services
- Less infrastructure maintenance effort
- Cost advantages with pay per use

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Texts (excl. quotes), presentation concept and selected logos/images/icons (excl. AWS icons): Copyright © 2022, Audi AG / XL2 GmbH. All rights reserved.

Additional challenges for smart manufacturing



Reusable



Agile and
highly adaptable



Self-optimizing

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Texts (excl. quotes), presentation concept and selected logos/images/icons (excl. AWS icons): Copyright © 2022, Audi AG / XL2 GmbH. All rights reserved.

“We are not in the business of being original. We are in the business of reusing things that work.”

Robert W. Bly

Copywriter

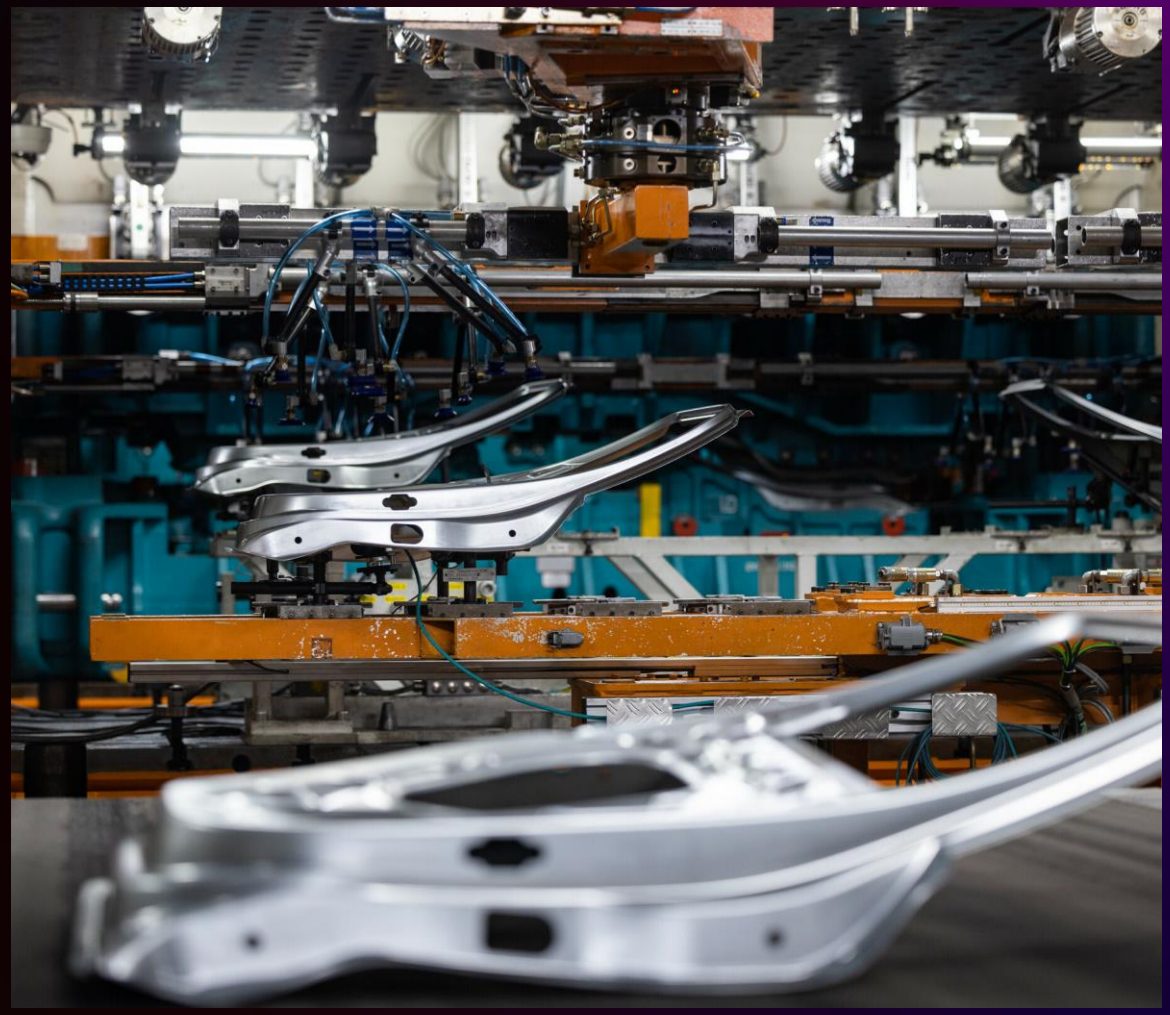
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Texts (excl. quotes), presentation concept and selected logos/images/icons (excl. AWS icons): Copyright © 2022, Audi AG / XL2 GmbH. All rights reserved.

Risk: Adding complexity to a complex system will make it even more complex

- Manufacturing system landscapes often contain silos with lots of **isolated solutions**
- Solutions for the **same problem** get **reinvented multiple times** and increase the overall production systems complexity

➡ Aim for reusable, mature solutions

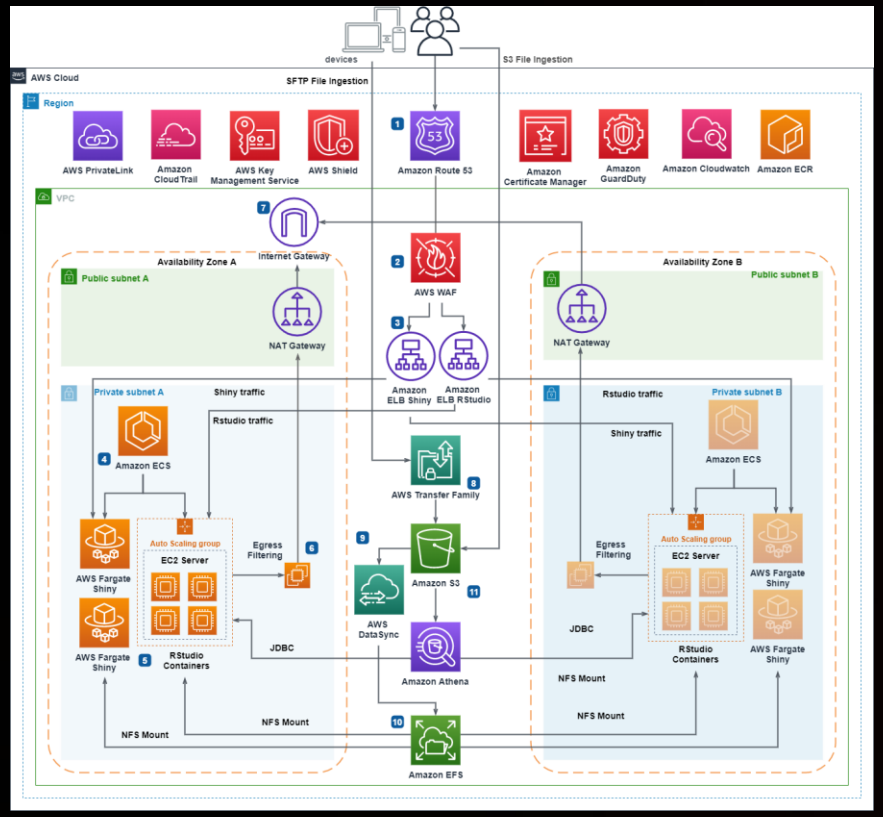


aws

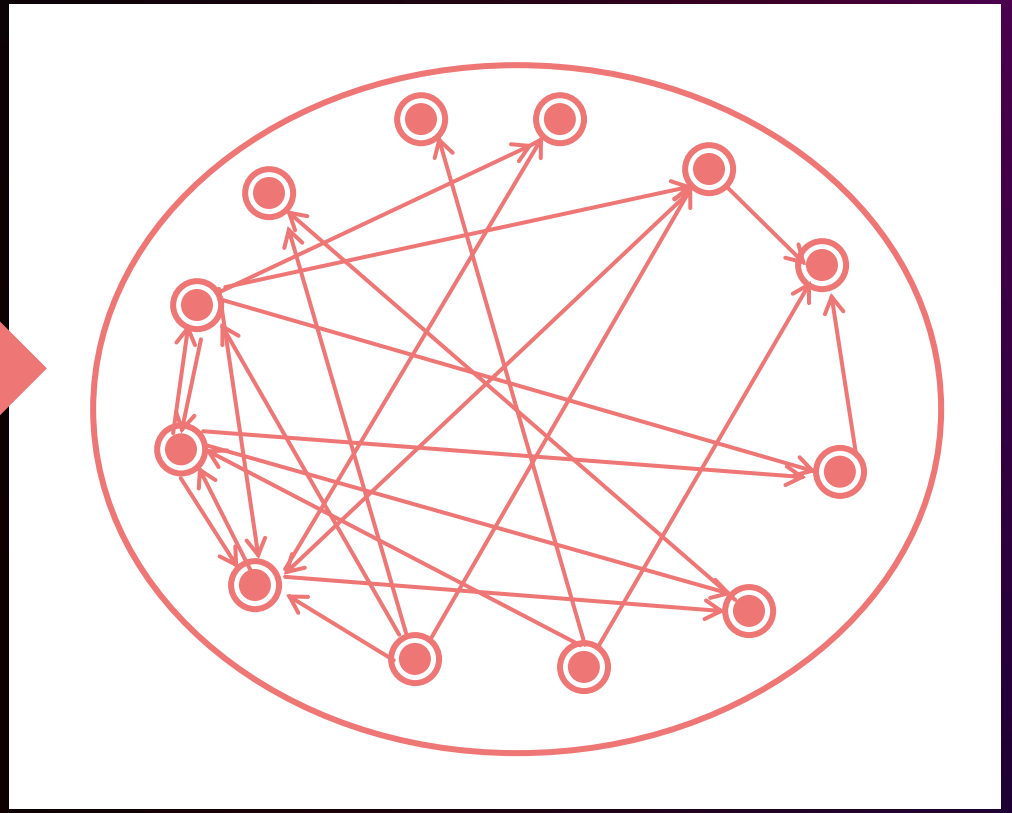
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Texts (excl. quotes), presentation concept and selected logos/images/icons (excl. AWS icons): Copyright © 2022, Audi AG / XL2 GmbH. All rights reserved.

Serverless solutions are at risk to become a complex dependency nightmare



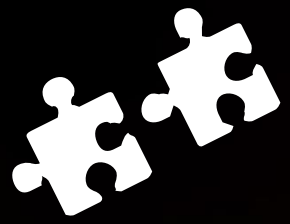
Arbitrary example from an AWS Blog post



“Big ball of mud” architecture

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Texts (excl. quotes), presentation concept and selected logos/images/icons (excl. AWS icons): Copyright © 2022, Audi AG / XL2 GmbH. All rights reserved.



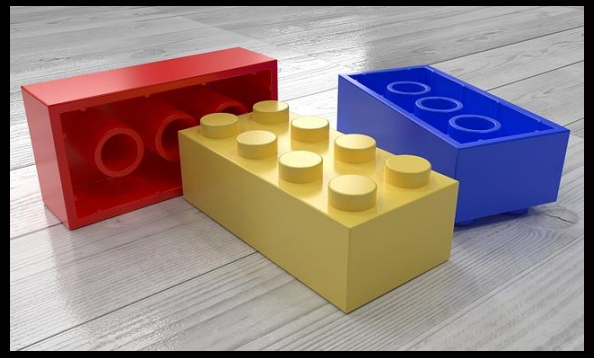
Reusable

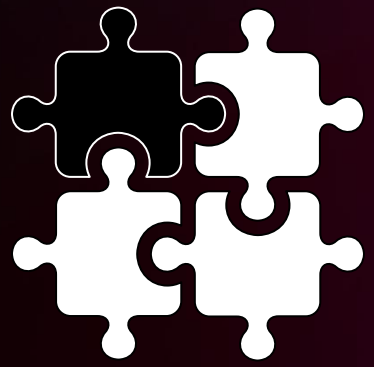
Design principles: Component-oriented design

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Texts (excl. quotes), presentation concept and selected logos/images/icons (excl. AWS icons): Copyright © 2022, Audi AG / XL2 GmbH. All rights reserved.

Following fundamental architecture principles is a prerequisite for reusability





Component-oriented design

- Separation of concerns
- Design by contract
- Information hiding
- Data sovereignty

Homogeneity

- Solve similar problems in similar ways
- But avoid unnecessary redundancy and duplicated code

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Texts (excl. quotes), presentation concept and selected logos/images/icons (excl. AWS icons): Copyright © 2022, Audi AG / XL2 GmbH. All rights reserved.

How to apply those principles to serverless applications

1. Group your AWS serverless resources into **microservices**. Cut them using the principles of **component-oriented design**, e.g., by separation of business concerns and capabilities

Examples for recurring concerns in production and logistics

- Order process
- Product identification
- Product position localization
- Task authorizations
- Process result collection
- Quality data identification
- Worker guidance processes
- Machine steering processes
- Reporting and analytics
- Technical monitoring
- Storing and keeping data for legal reasons

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Texts (excl. quotes), presentation concept and selected logos/images/icons (excl. AWS icons): Copyright © 2022, Audi AG / XL2 GmbH. All rights reserved.

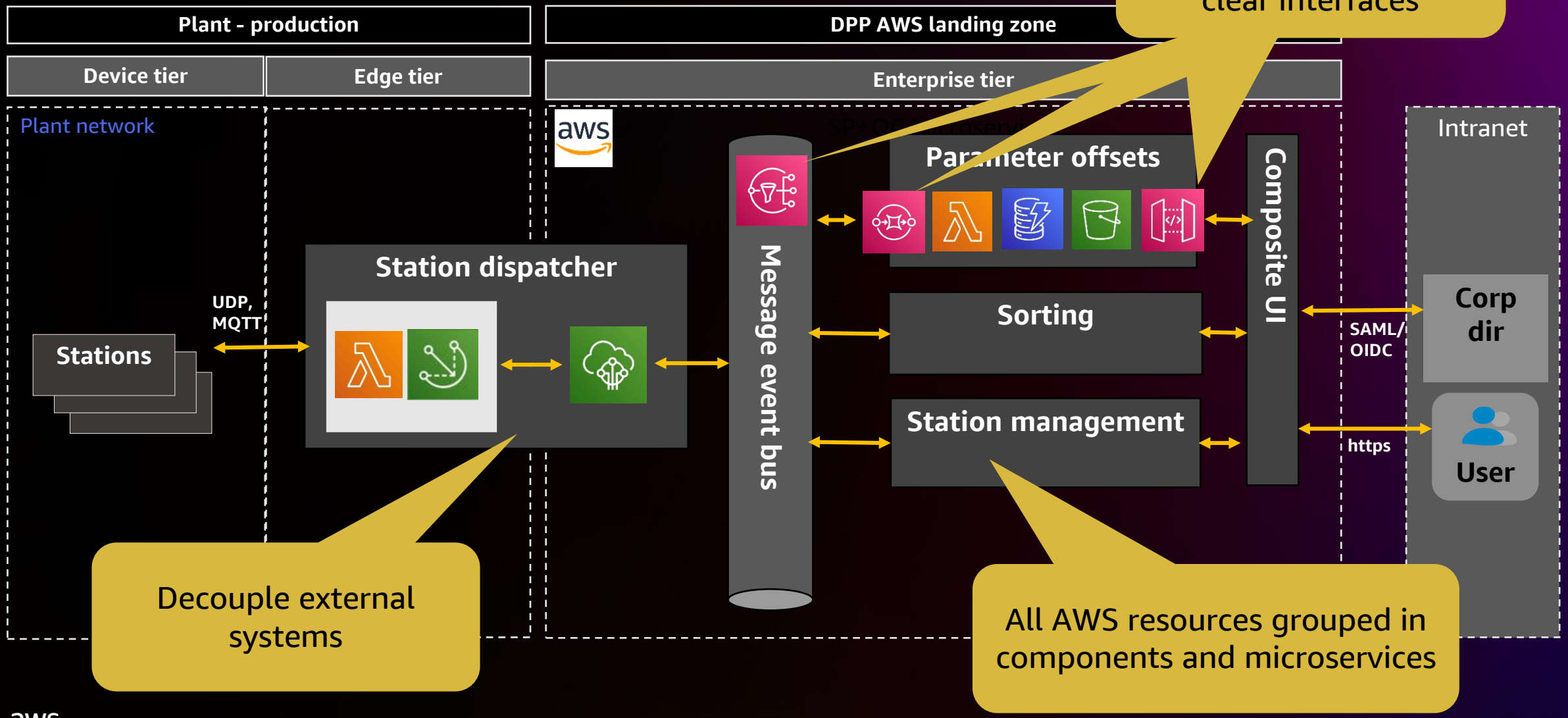
How to apply those principles to serverless applications

2. Decouple the microservices using **AWS application integration services** (e.g., with Amazon SNS, Amazon SQS, or Amazon API Gateway)
3. Provide **clear component interface specifications** (e.g., with OpenAPI or AsyncAPI)
4. Decouple **dependencies to external systems** (e.g., MES systems or shop-floor stations)
5. Apply **well-known serverless design patterns** (e.g., the SNS/SQS fan out pattern)

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Texts (excl. quotes), presentation concept and selected logos/images/icons (excl. AWS icons): Copyright © 2022, Audi AG / XL2 GmbH. All rights reserved.

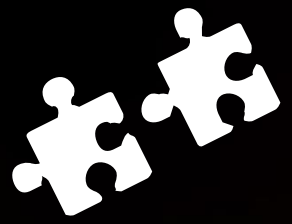
Example: Parameter service

PARTIAL ARCHITECTURE VIEW



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Texts (excl. quotes), presentation concept and selected logos/images/icons (excl. AWS icons): Copyright © 2022, Audi AG / XL2 GmbH. All rights reserved.



Reusable

Design principles: Service stores

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Texts (excl. quotes), presentation concept and selected logos/images/icons (excl. AWS icons): Copyright © 2022, Audi AG / XL2 GmbH. All rights reserved.

Learn software reuse concepts from app stores

- Apps usually solve a single use case
- Apps are part of an ecosystem

Business departments would like to have something like this for their smart manufacturing plants

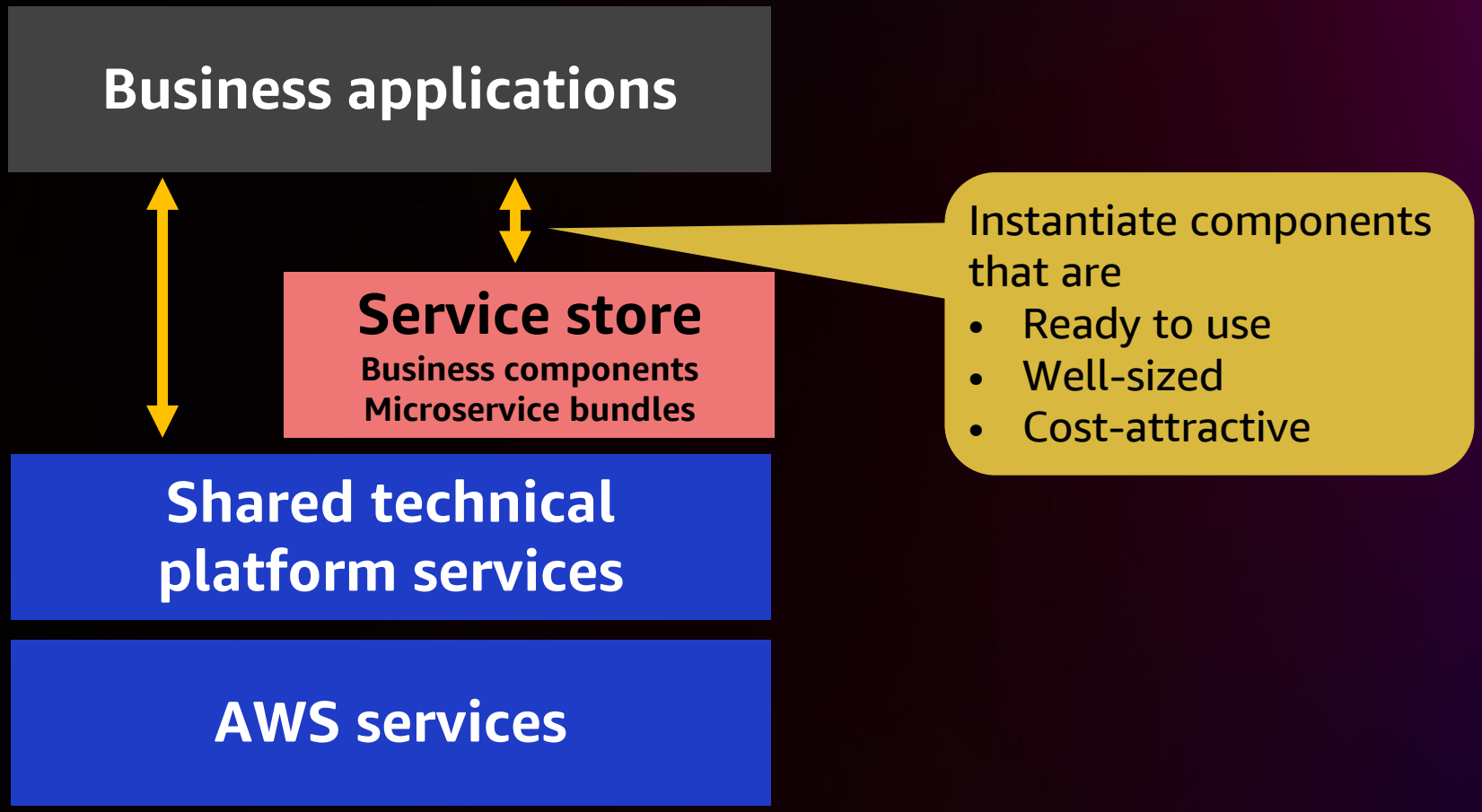


aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Texts (excl. quotes), presentation concept and selected logos/images/icons (excl. AWS icons): Copyright © 2022, Audi AG / XL2 GmbH. All rights reserved.

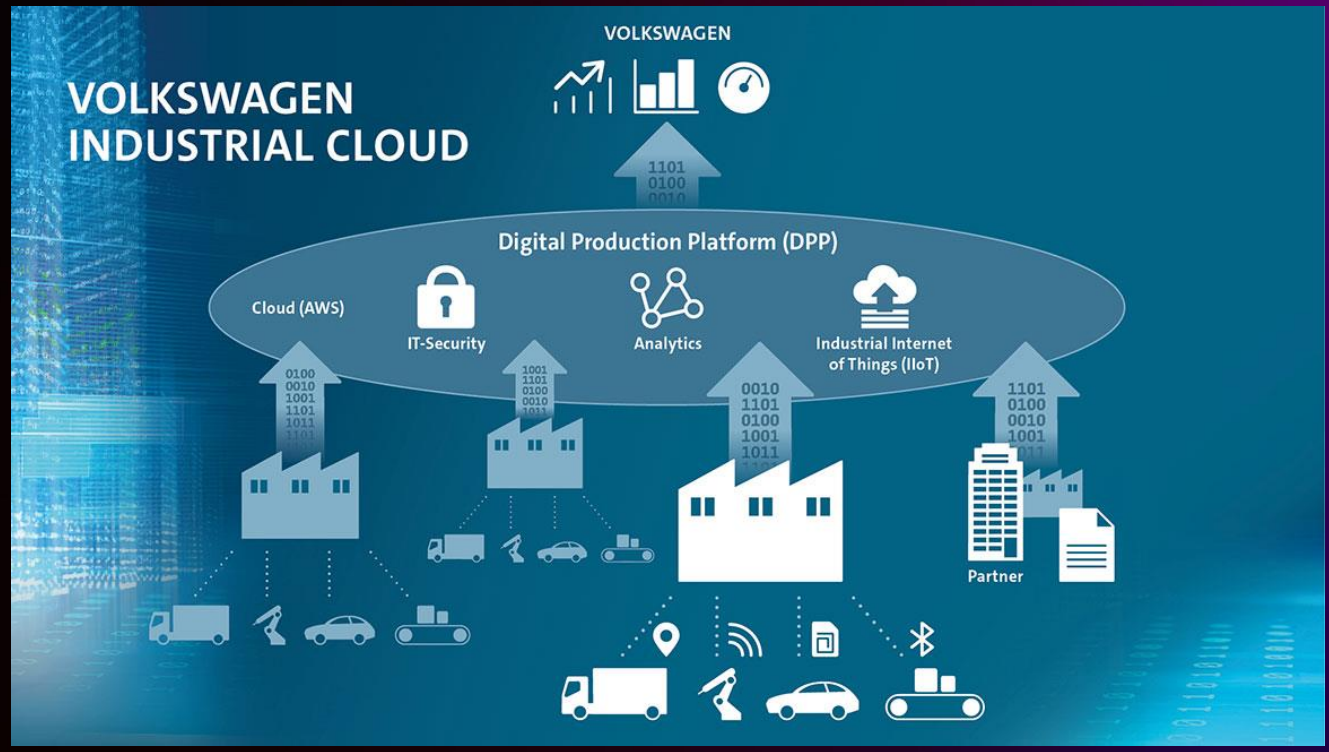
Idea: Introduce service stores as company catalog for reusable software bundles



aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Texts (excl. quotes), presentation concept and selected logos/images/icons (excl. AWS icons): Copyright © 2022, Audi AG / XL2 GmbH. All rights reserved.

Example: The Volkswagen DPP

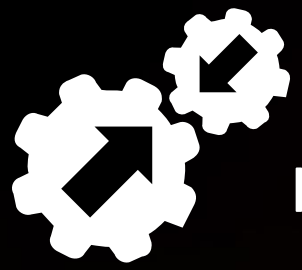
- **DPP = Digital Production Platform**
- Provides not only AWS services and a landing zone, but also reusable services on top of AWS
- **DPP Quickstarter Kit** as evolving catalog for reusable business solutions
- **Parameter service already reused for several use cases** (e.g., rear axle setting, headlights, battery testers)



Source: https://www.volkswagenag.com/de/news/stories/2019/03/volkswagen-industrial-cloud.html

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Texts (excl. quotes), presentation concept and selected logos/images/icons (excl. AWS icons): Copyright © 2022, Audi AG / XL2 GmbH. All rights reserved.



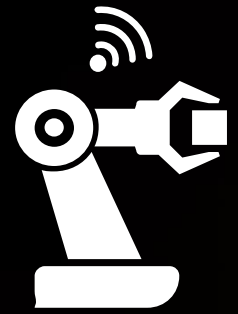
Highly adaptable

Design principles: Elastic data models

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

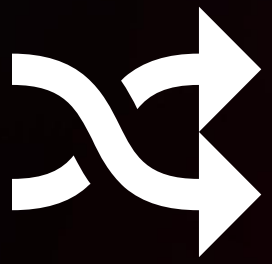
Texts (excl. quotes), presentation concept and selected logos/images/icons (excl. AWS icons): Copyright © 2022, Audi AG / XL2 GmbH. All rights reserved.

Why do smart manufacturing systems need to be highly adaptable?



Brownfield scenarios

E.g., support for different communication protocols (UDP, MQTT, OPC/UA, …)



Production line changes

E.g., support for new model series or plant-specific changes



Agility

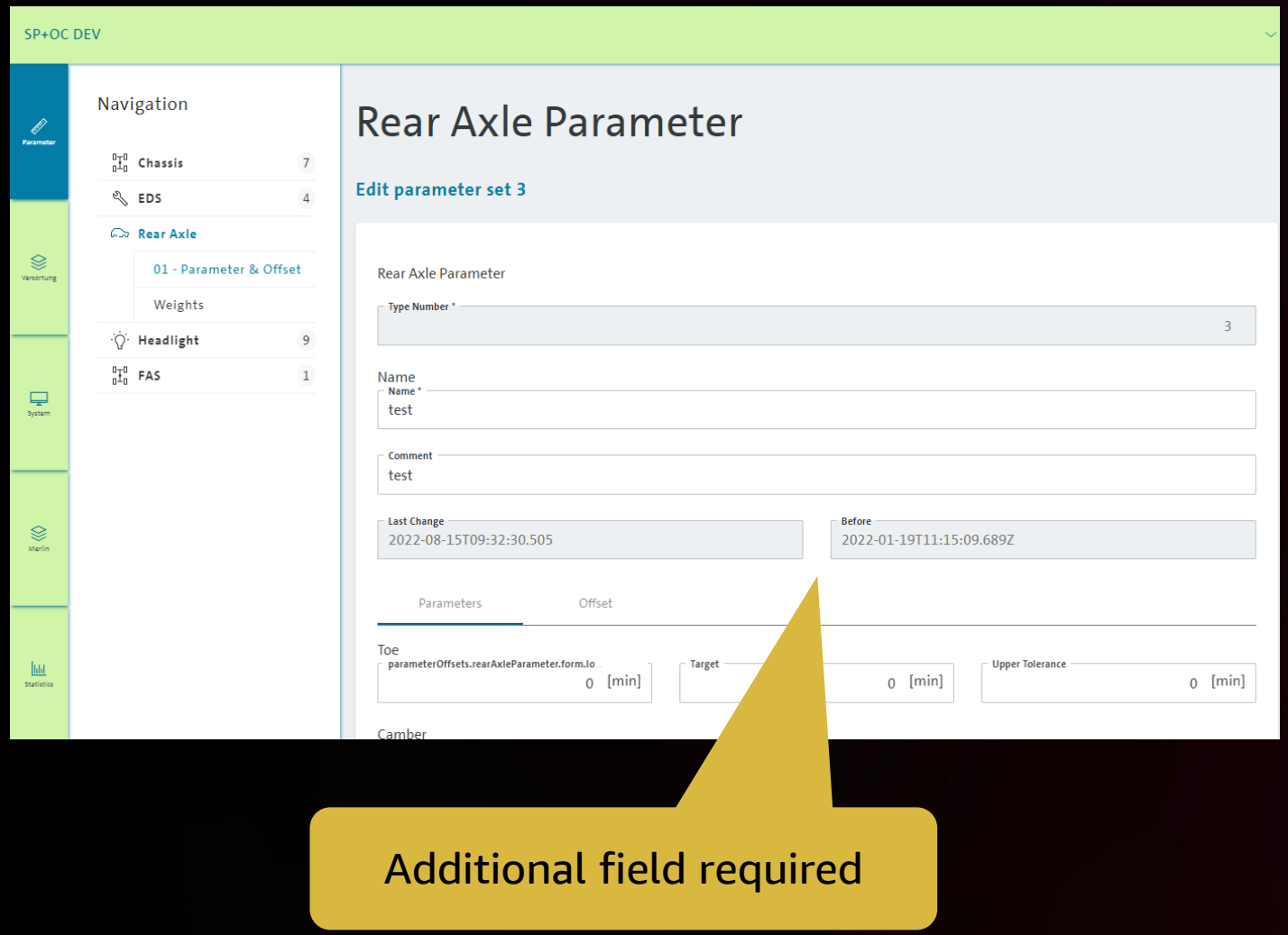
E.g., quick reaction on supply chain disruptions or movement to digital processes

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Texts (excl. quotes), presentation concept and selected logos/images/icons (excl. AWS icons): Copyright © 2022, Audi AG / XL2 GmbH. All rights reserved.

Example: Add an additional field to an entity



Additional field required

Requirements

Business department wants to add an attribute to a parameter set

Old data needs to be historized due to legal requirements

Deployment to all microservices during system runtime, without a software change

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Texts (excl. quotes), presentation concept and selected logos/images/icons (excl. AWS icons): Copyright © 2022, Audi AG / XL2 GmbH. All rights reserved.

Design Principle: Elasticity in data models

Elastic data model

Adaptable data model for structured data that can be modified during the runtime of an application without changes to database schemas or business logic source code (e.g., for input data validation)

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Texts (excl. quotes), presentation concept and selected logos/images/icons (excl. AWS icons): Copyright © 2022, Audi AG / XL2 GmbH. All rights reserved.

Step 1

Define JSON schemas for your structured data

Parameter set JSON schema

```
{
    "$schema": "http://json-schema.org/draft-07
    "type": "object",
    "additionalProperties": false,
    "properties": {
        "comment": {
            "type": "string",
            "default": "",
            "maxLength": 500
        },
        "name": {
            "type": "string",
            "title": "The name schema",
            "default": "",
            "examples": [],
            "minLength": 1,
            "maxLength": 80
        },
        "toe": {
            "description": "{ \"label.de\" : \"Spur:\", \"label.en\" : \"toe:\" }",
            "type": "object",
            "properties": {
                "toeRearLowerTolerance": {
                    "type": ["number","null"],
                    "minimum": -99.99,
                    "maximum": 99.99,
                    "multipleOf": 0.01
                },
                "toeRearTarget": {
                    "type": ["number","null"],
                    "minimum": -99.99,
                    "maximum": 99.99,
                    "multipleOf": 0.01
                }
```

Use JSON objects to store your records

Include metadata in the schema, and use it for input data validation of the objects

Propagate the JSON schemas to all microservices handling this record type, e.g., with SNS

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Texts (excl. quotes), presentation concept and selected logos/images/icons (excl. AWS icons): Copyright © 2022, Audi AG / XL2 GmbH. All rights reserved.

Step 2

Generate UI forms from the JSON schemas, targeting a template engine

Parameter set JSON schema



Code generator

Parameter UI layout JSON



Template engine:

https://**FORMLY**.dev

Library for Angular

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Texts (excl. quotes), presentation concept and selected logos/images/icons (excl. AWS icons): Copyright © 2022, Audi AG / XL2 GmbH. All rights reserved.

Step 3

Embed dynamically rendered UI forms in your application

Layout JSON

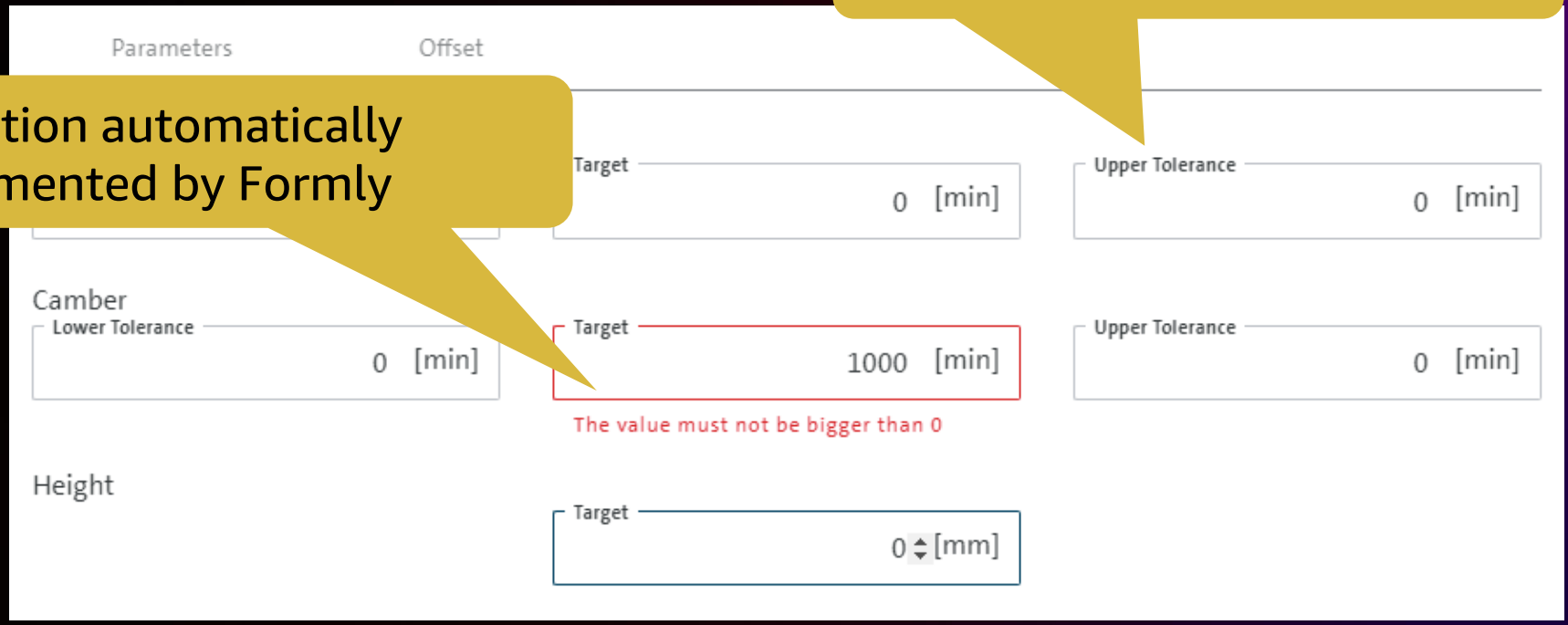
```
[
  {
    "fieldGroupClassName": "display-flex", // <-- wrapper for multiple form fields
    "fieldGroup": [
      {
        "className": "section-label vertical-align-middle",
        "template": "form.toe" // <-- translation placeholder for the section label
      },
      {
        "key": "toe.specification",
        "type": "input",
        "templateOptions": {
            "className": "flex-1"
            // other template options
        }
      },
      {
        "key": "toe.tolerance",
        "type": "input",
        "templateOptions": {
            "className": "flex-1"
            // other template options
        }
      },
      {} // <-- empty cell in the table
    ]
```

Dynamically at runtime by formly.dev



Validation automatically implemented by Formly

Labels in i18n config file



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Texts (excl. quotes), presentation concept and selected logos/images/icons (excl. AWS icons): Copyright © 2022, Audi AG / XL2 GmbH. All rights reserved.

Elastic data model example with JSON schemas

Pros

- Highly configurable UI, deployable during system runtime
- Supports data model versioning
- Works well with serverless services like Amazon DynamoDB

Cons

- Still needs testing before go live
- Limitations for queries on JSON objects
- Works not so well if you need an RDS

Application areas

- Data records like station parameters, sensor data, logistic tracing records, …

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Texts (excl. quotes), presentation concept and selected logos/images/icons (excl. AWS icons): Copyright © 2022, Audi AG / XL2 GmbH. All rights reserved.



Self-optimizing

Design principles: Closed feedback loops

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Texts (excl. quotes), presentation concept and selected logos/images/icons (excl. AWS icons): Copyright © 2022, Audi AG / XL2 GmbH. All rights reserved.

Goal: Self-optimizing production systems

- **Optimize** cost, quality, and delivery time **in complex environments**
- Improve the resilience by **automatically handling unforeseen disturbances** within supply chains, machines, and processes

→ Aim for self-optimizing systems to **improve the competitiveness** of manufacturing companies

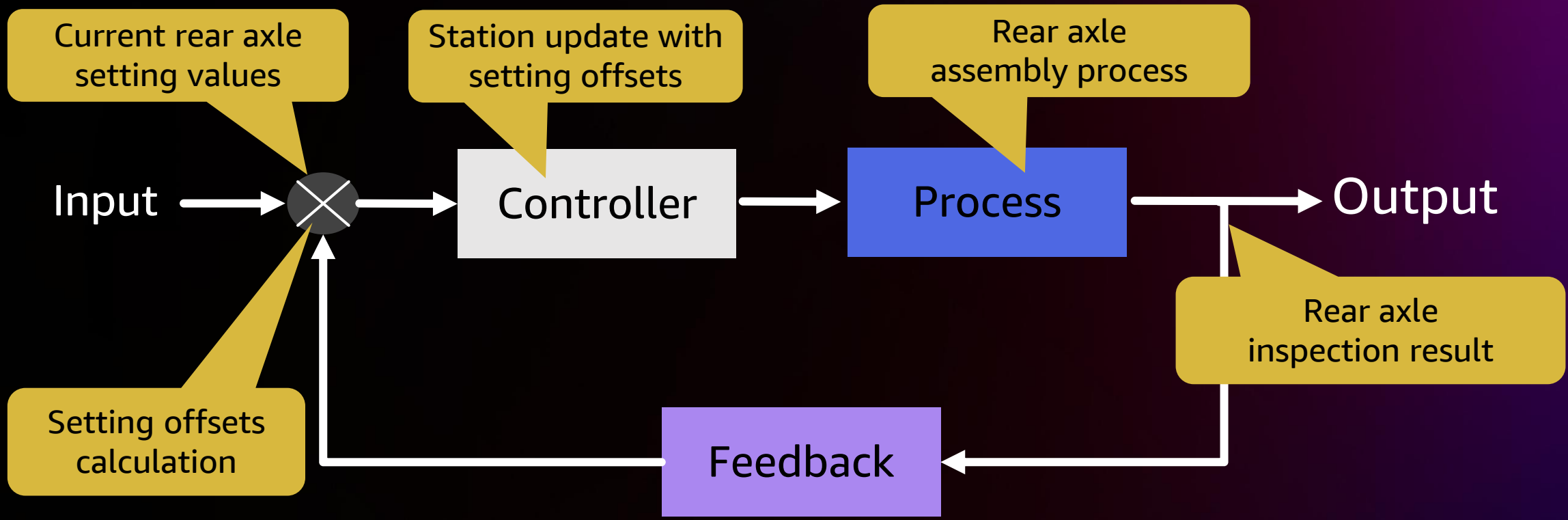


aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Texts (excl. quotes), presentation concept and selected logos/images/icons (excl. AWS icons): Copyright © 2022, Audi AG / XL2 GmbH. All rights reserved.

Principle of closed feedback loops



aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Texts (excl. quotes), presentation concept and selected logos/images/icons (excl. AWS icons): Copyright © 2022, Audi AG / XL2 GmbH. All rights reserved.

Design principle: Event-driven architectures as foundation for closed feedback loops



AWS serverless application integration services like Amazon SNS, Amazon SQS, and Amazon EventBridge allow an easy **setup of event channels**



Use **publish/subscribe** for communication between the microservices **in your application**



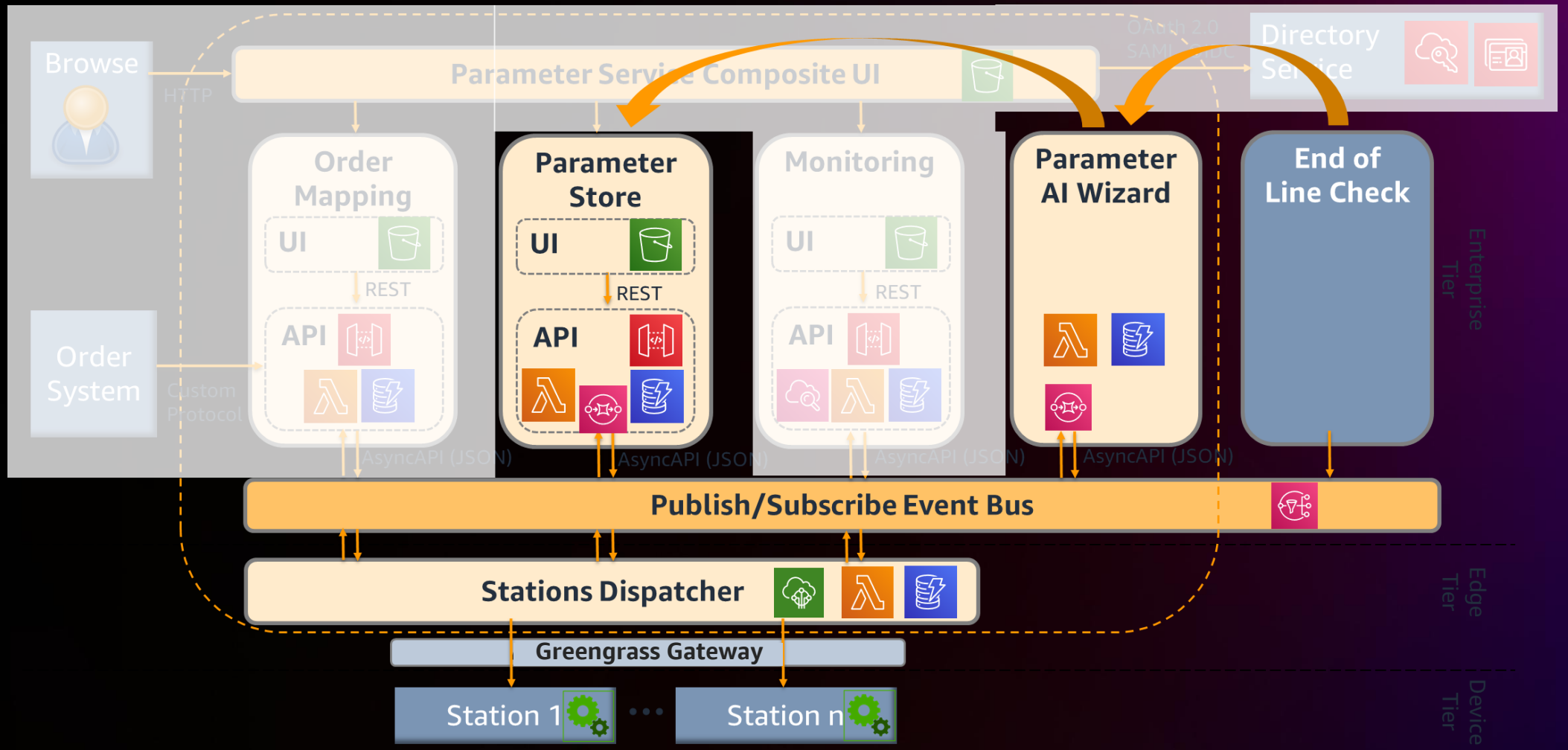
Apply the **event-driven pattern** also to **connect to external systems** in a loosely coupled way

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Texts (excl. quotes), presentation concept and selected logos/images/icons (excl. AWS icons): Copyright © 2022, Audi AG / XL2 GmbH. All rights reserved.

Example: Closed feedback loop in the parameter service



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

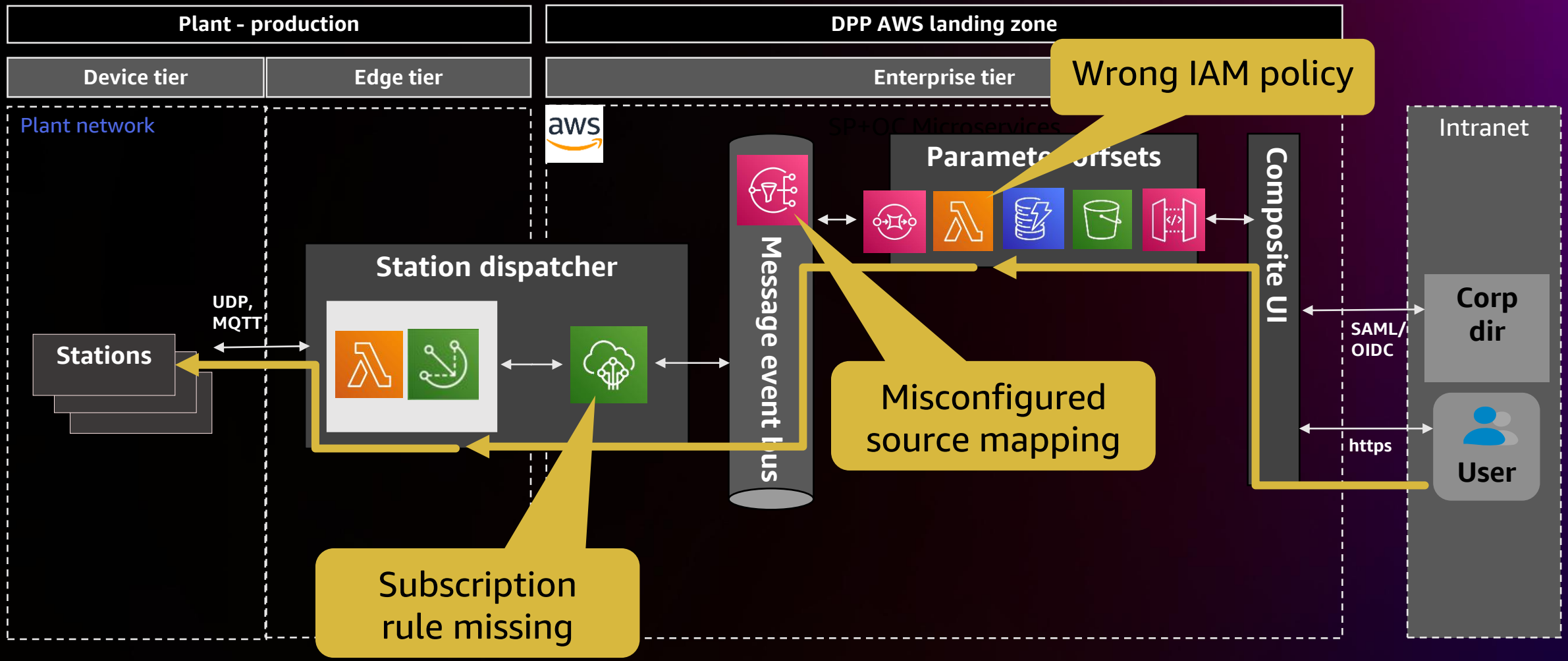
Texts (excl. quotes), presentation concept and selected logos/images/icons (excl. AWS icons): Copyright © 2022, Audi AG / XL2 GmbH. All rights reserved.

Development principles for serverless projects

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Texts (excl. quotes), presentation concept and selected logos/images/icons (excl. AWS icons): Copyright © 2022, Audi AG / XL2 GmbH. All rights reserved.

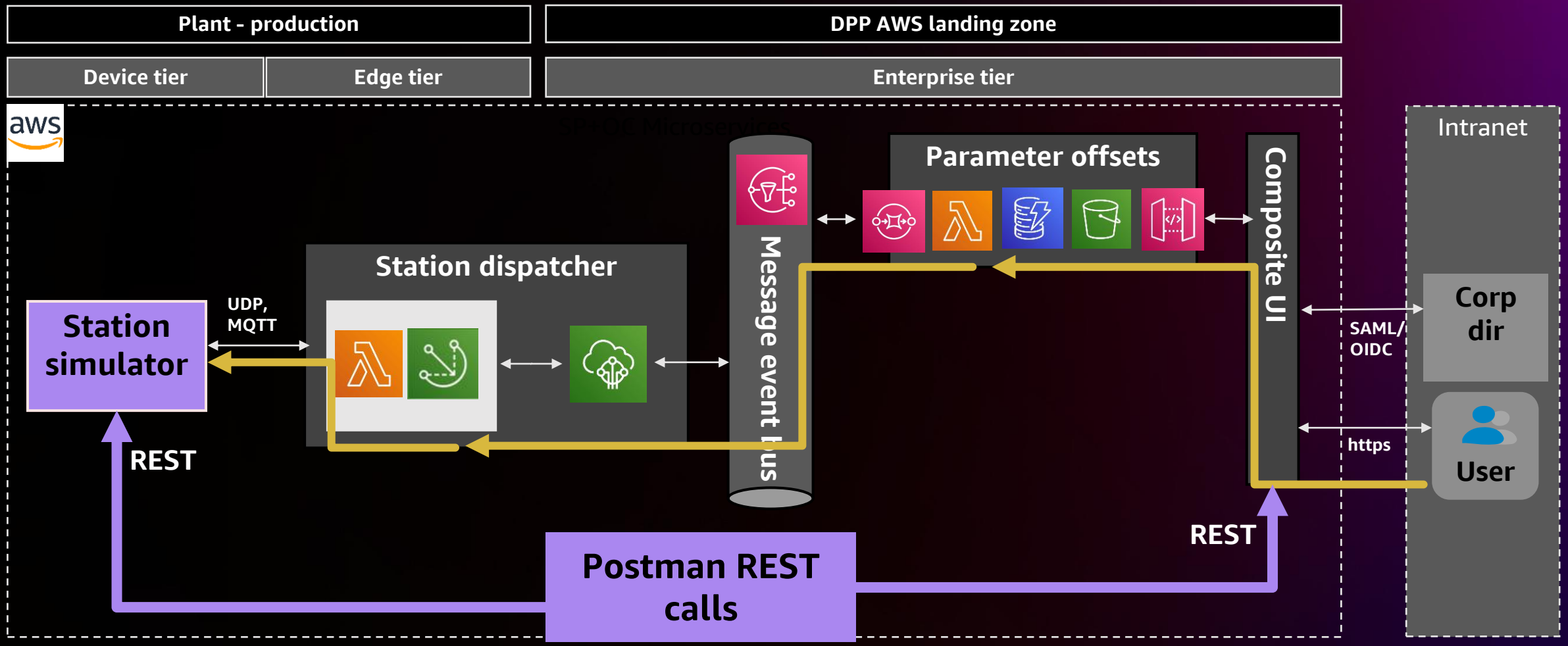
Testing the user story: “Send parameter to a station”



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

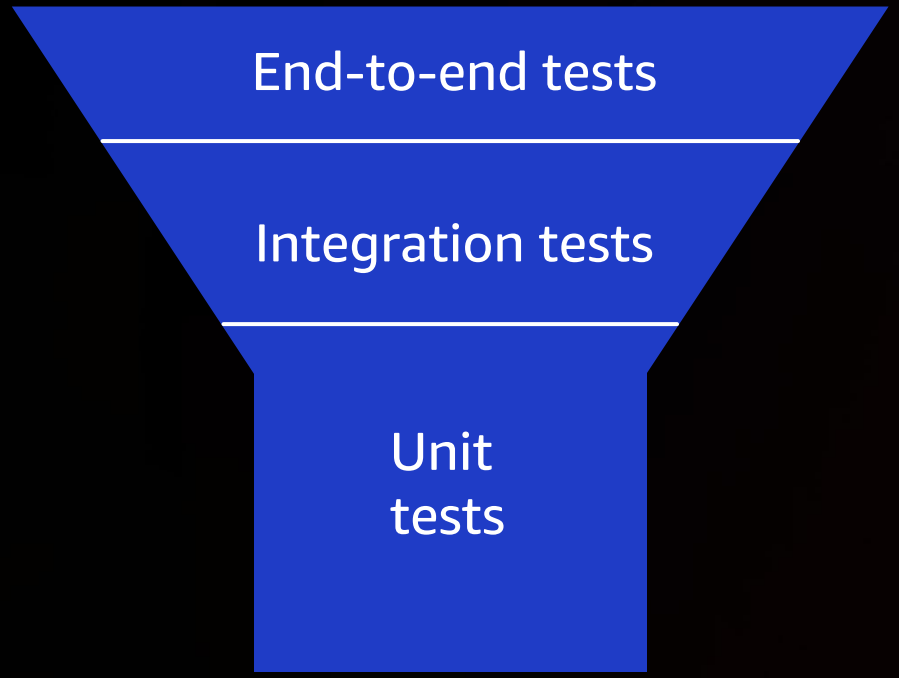
Texts (excl. quotes), presentation concept and selected logos/images/icons (excl. AWS icons): Copyright © 2022, Audi AG / XL2 GmbH. All rights reserved.

Testing the user story: “Send parameter to a station”



aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Texts (excl. quotes), presentation concept and selected logos/images/icons (excl. AWS icons): Copyright © 2022, Audi AG / XL2 GmbH. All rights reserved.

Development principle: End-to-end test over local testing



Inverted testing pyramid

- Functionality of serverless applications can break due to errors in security and resource policies
- Local developer tests can be complicated as not all cloud services can be mocked

Smart serverless applications should have automated end-to-end-tests running in the cloud environment

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Texts (excl. quotes), presentation concept and selected logos/images/icons (excl. AWS icons): Copyright © 2022, Audi AG / XL2 GmbH. All rights reserved.

Development principle

Infrastructure as code development == Source code development



- Use tools like AWS CDK that provide a programming-like development experience for infrastructure as code. Plain AWS CloudFormation could get cumbersome.



- Ensure the quality of infrastructure as code as strict as for the rest of your code. Apply the same principles for testing, code reviews, cleanliness, modularization, …

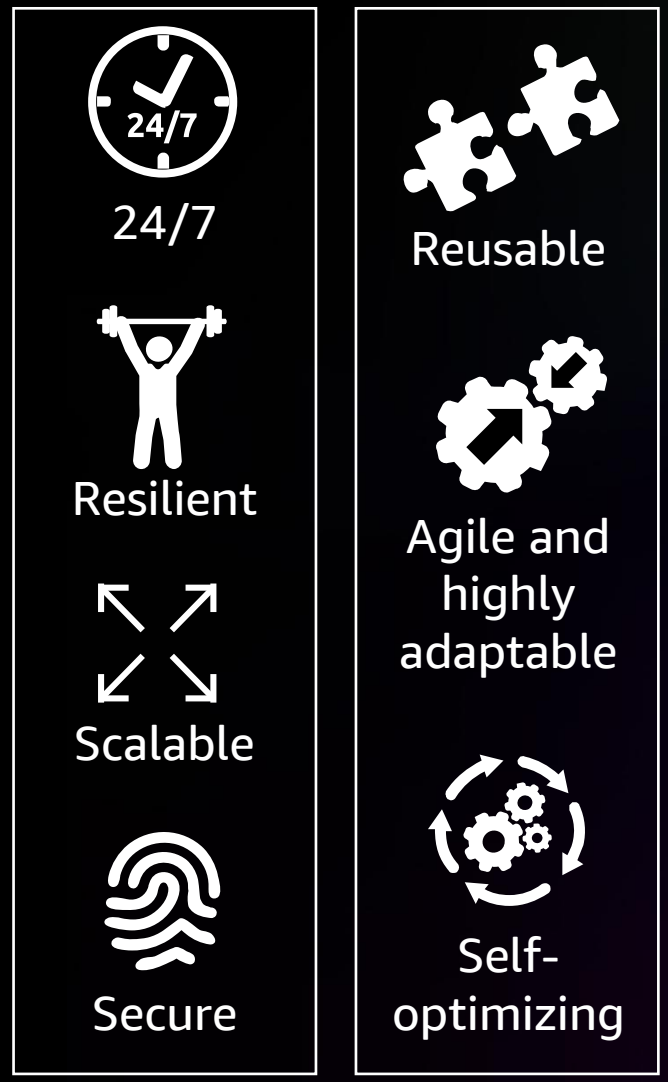
aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Texts (excl. quotes), presentation concept and selected logos/images/icons (excl. AWS icons): Copyright © 2022, Audi AG / XL2 GmbH. All rights reserved.

Summary

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Texts (excl. quotes), presentation concept and selected logos/images/icons (excl. AWS icons): Copyright © 2022, Audi AG / XL2 GmbH. All rights reserved.

Smart serverless for smart manufacturing



Principles and patterns

1. Component-oriented design and homogeneity
2. Service stores containing reusable serverless microservice bundles
3. Elasticity of data models
4. Event-driven architecture
5. Closed feedback loops for self-optimizing systems
6. End-to-end tests over local testing
7. Infrastructure-as-code development == source code development

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Texts (excl. quotes), presentation concept and selected logos/images/icons (excl. AWS icons): Copyright © 2022, Audi AG / XL2 GmbH. All rights reserved.

Thank you!

Jens Busch
jens1.busch@audi.de

Markus Großmann
markus.grossmann@xl2.de
linkedin.com/in/markusgrossmann



Please complete the session survey in the **mobile app**

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Texts (excl. quotes), presentation concept and selected logos/images/icons (excl. AWS icons): Copyright © 2022, Audi AG / XL2 GmbH. All rights reserved.