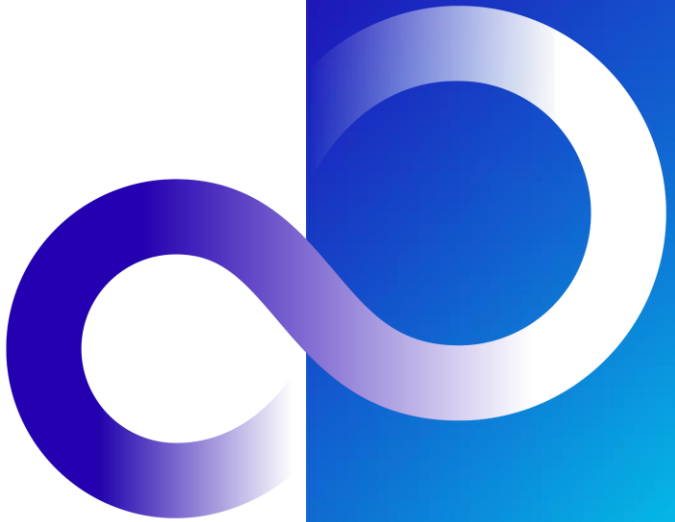
Edge Service Enabling Platform Fall 2023 Update

Akraino Fall Summit 2023

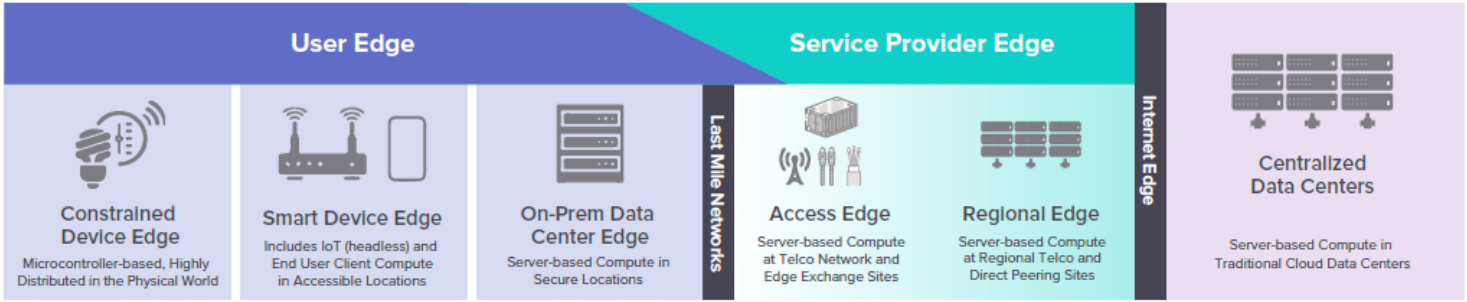
Colin Peters, Fujitsu Limited



What's the Problem?

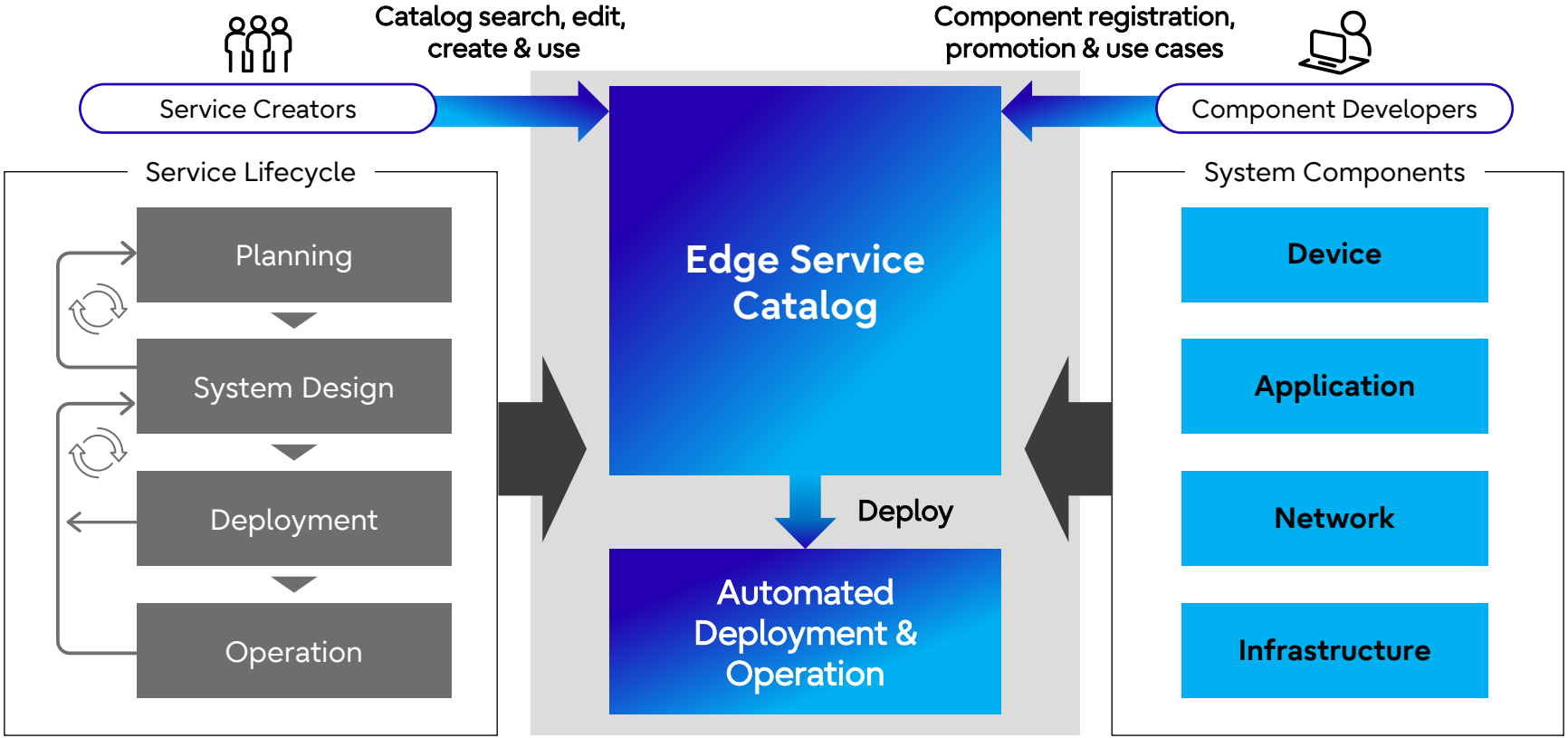
FUJITSU

- Edge services have great potential, but…
  - Implementing an edge service is *hard*
  - Many technologies, many areas of expertise
  - Rapid evolution and change
- The Edge Service Enabling Platform is here to fix that



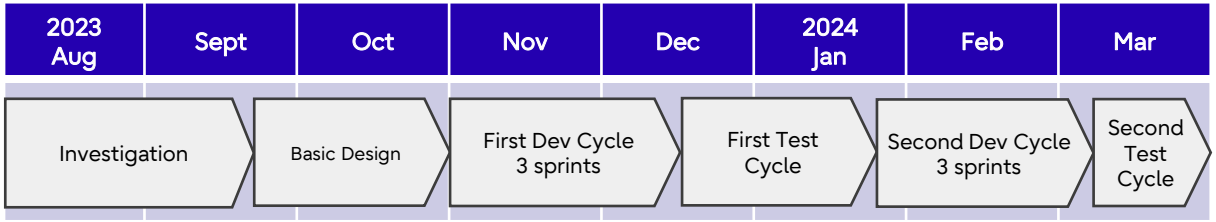
LF Edge Taxonomy from Sharpening the Edge: Overview of the LF Edge Taxonomy and Framework
https://www.lfedge.org/resources/publications/

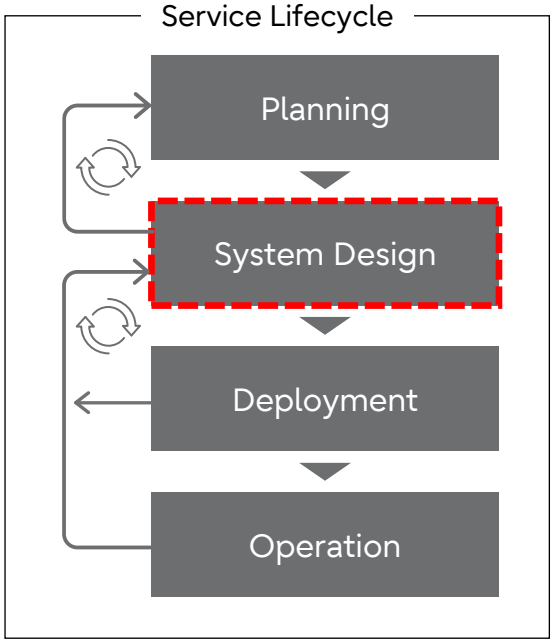
Edge Service Enabling Platform (ESPF)



What We're Doing in 2023

- Prototyping an open tool which will help make designing edge services easier
  - Improve on existing tooling for edge service designers in a substantial way
  - Provide an open-source implementation
  - Have something usable in a short (<1year) time frame





Focus first on the service/system design part of the lifecycle

What Tooling, And Why

FUJITSU

- How to make edge service design easier?
  - Abstract away complexity
    - Information hiding (ESPF “components”)
- Let’s use TOSCA
  - Can model all the things
  - TOSCA’s already established as a standard
- Eclipse Winery is a good TOSCA tool, let’s build on that
  - Open-source, so we can extend it as we need

OASIS TOSCA

FUJITSU

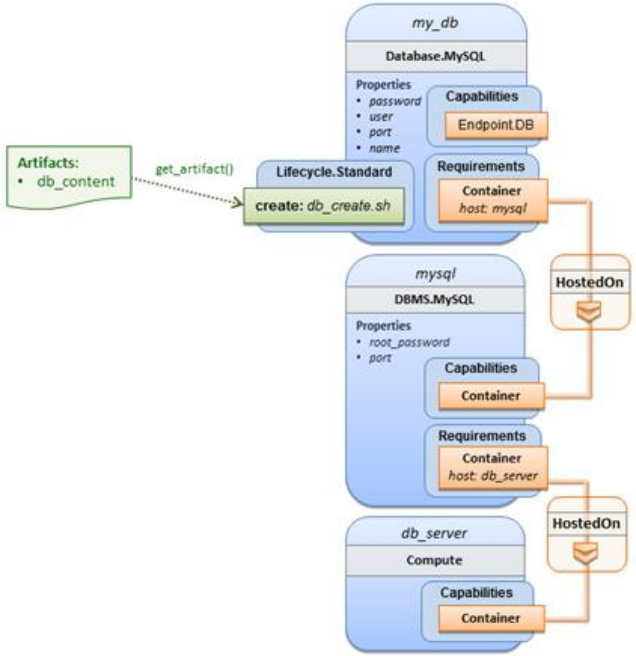
- Why TOSCA?
  - TOSCA “provides a language to describe service components and their relationships”
    - https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=tosca
  - Several open-source implementations, including ONAP
  - Active and relatively mature
  - High-level, abstract, and flexible

```
tosca_definitions_version: tosca_simple_yaml_1_3tosca_simple_yaml_1_3

description: Template for deploying a single server with predefined properties.

topology_template:
  node_templates:
    db_server:
      type: tosca.nodes.Compute
      capabilities:
        # Host container properties
        host:
         properties:
           num_cpus: 1
           disk_size: 10 GB
           mem_size: 4096 MB
        # Guest Operating System properties
        os:
          properties:
            # host Operating System image properties
            architecture: x86_64
            type: linux
            distribution: rhel
            version: 6.5
```

An example of the TOSCA language in YAML (from TOSCA Simple Profile in YAML Version 1.3)

TOSCA Model

- TOSCA's model & terminology
  - A *service template* contains a *topology template*, which contains *node templates* (or just "nodes")
  - Nodes are connected by *relationships* which join node *requirements* to node *capabilities*
  - What *requirements*, *capabilities*, and *properties* a node has is defined by their node type
    - Similarly for relationships and their types
- TOSCA also has *artifacts*, *interfaces*, *operations*, *data types*...



Representation of a database, the DBMS that hosts it, and the compute node that hosts the DBMS, as objects in TOSCA (from [TOSCA Simple Profile in YAML Version 1.3])

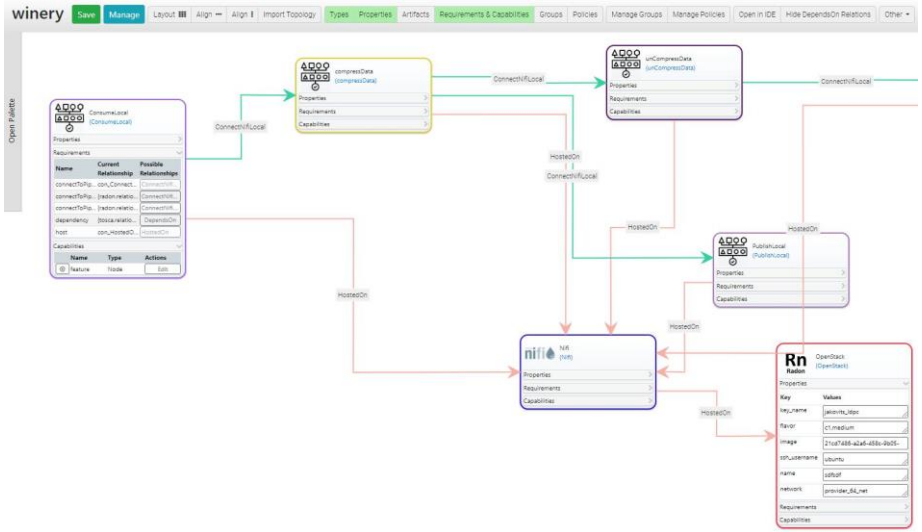
Choosing TOSCA & Eclipse Winery

FUJITSU

- TOSCA is very close to what we want as a data model
  - Already established, open standard
  - Has many of the modeling features we need
  - Has something like the abstractions we want (node substitution), but…
    - Not very well described in the standard
    - Not widely implemented
    - Choice of substitution is performed by orchestrator based on filters
- With just a little work, we could add the abstraction we want
- Eclipse Winery has the pieces we need to build on…

Eclipse Winery

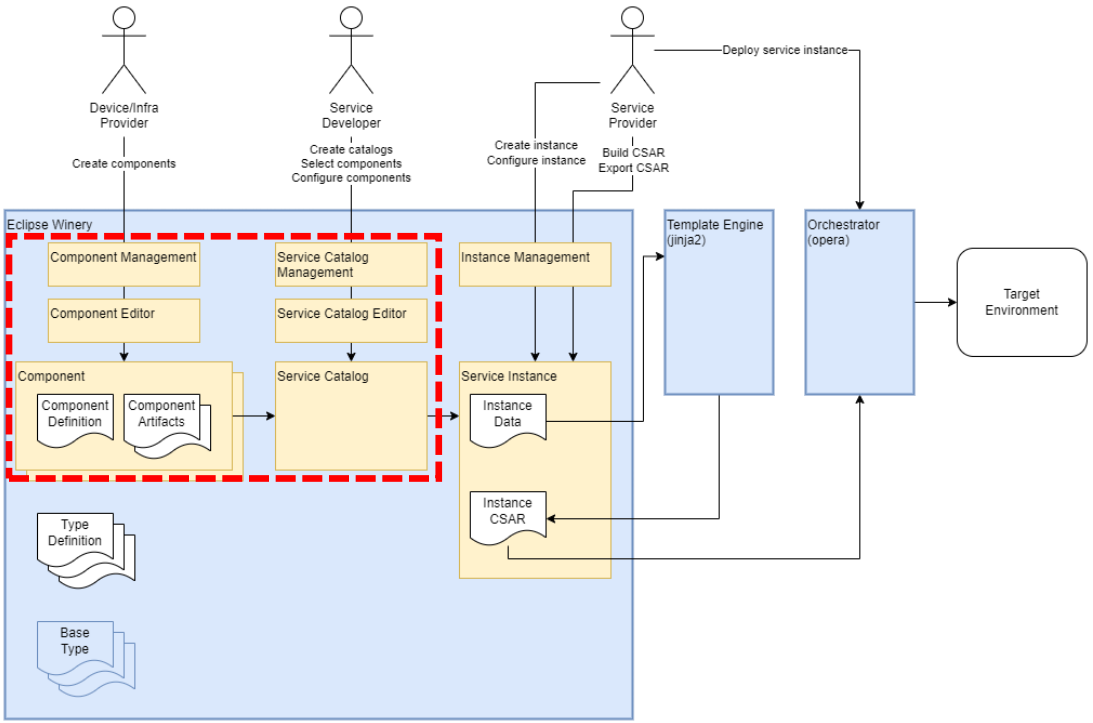
- Building on Winery
  - Open-source Java & Typescript GUI editor for service design
  - Full-featured interactive topology editing
  - Uses TOSCA language internally
  - Import & export TOSCA



A screenshot of Eclipse Winery's service topology editor

ESPF 2023 Architecture

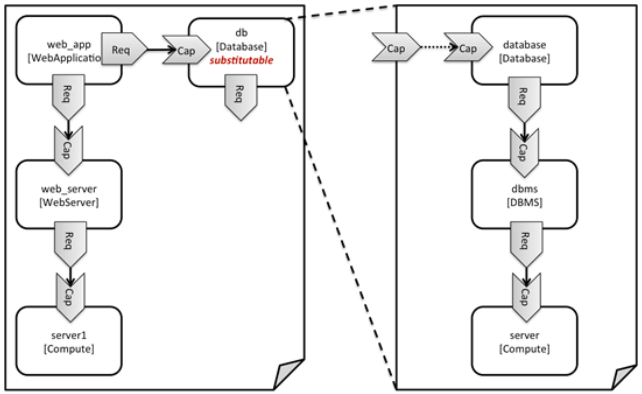
- Eclipse Winery is our front end
  - Platform features like user management come later
- Functions for service design first
  - Instances and deployment come later
- Additions:
  - Component support
  - Service catalogs



Mid-term architecture/process flow for ESPF (instance manager and orchestration integration is future work)

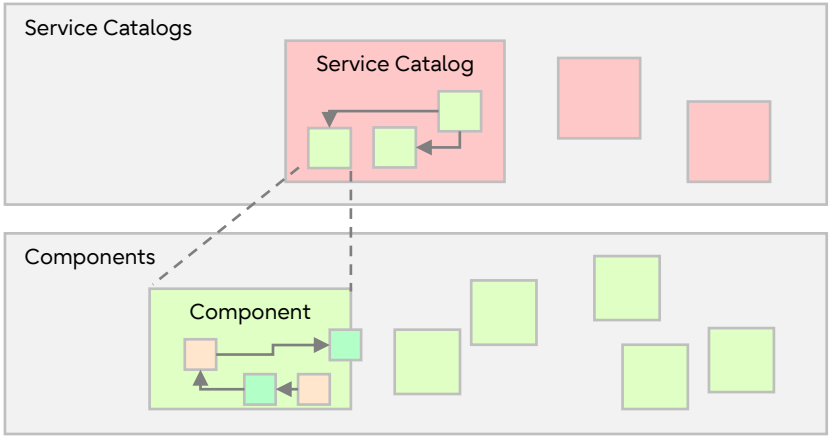
Development Items

- Components
  - encapsulate complex groups of nodes and relationships
  - i.e., abstraction



A "database" component which contains a more detailed representation of the nodes and relationships within it (from TOSCA Simple Profile in YAML Version 1.3)

- Service Catalogs
  - Initially, just a service template that contains components
  - Specify components directly



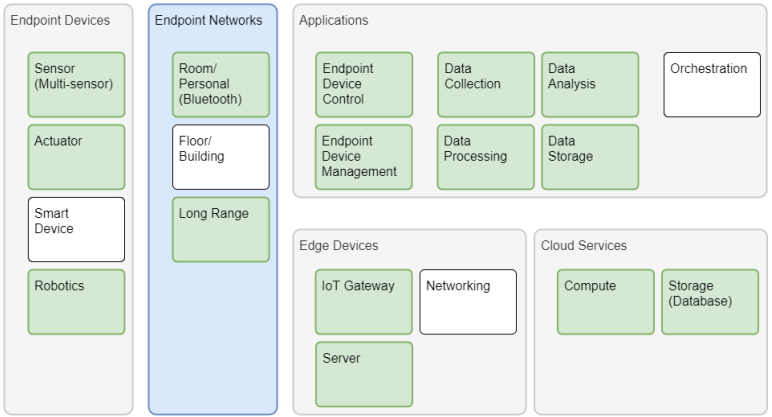
Future Development

FUJITSU

- Ideas for 2024
  - Deploying services
    - We're not focused on the deployment part of the lifecycle this year
    - There are lots of existing projects even in open-source
    - Orchestrator agnostic
  - Usability improvements
    - Many small improvements to make
    - Search & filtering
  - Component connection UX
    - One major source of complexity is connections between nodes
    - Simplify, document, and automate
- More?

Long-Term Vision

- Extend the available abstraction layers
  - Major functional groupings
  - Technology categories
- Simplify component selection
- Design automation
  - Suggest components
  - Automate connections
- Conversational UI
- Full platform!



A mockup of an abstract component selection UI

ESPF Further Reading

FUJITSU

- Akraino Blueprint Wiki:
  - https://wiki.akraino.org/display/AK/Edge+Service+Enabling+Platform
- Latest Design Notes:
  - https://wiki.akraino.org/display/AK/2023+Service+Design+Tool+Prototype+Implementation
- “Edge for Everybody” talk from OSS Japan 2022:
  - https://youtu.be/zAVkNyYN8jI?t=1331

Thank you