Digital Twin as a Service Software Platform

🏭 ↔️ 👥

DTaaS

Prasad Talasila

prasad.talasila@ece.au.dk

PRESENTATION OUTLINE

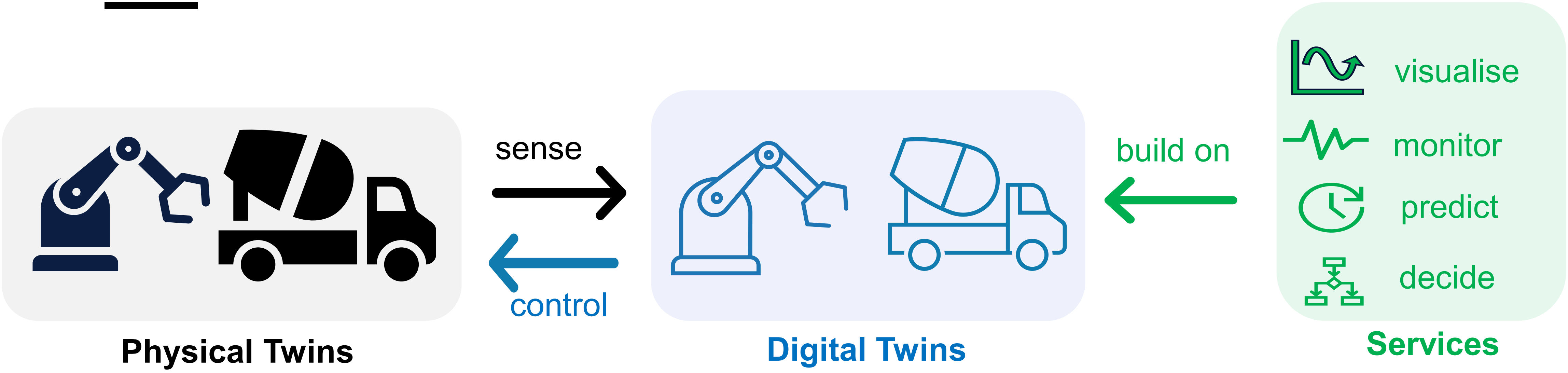
1) What is inside a Digital Twin?
2) Who are Users?
3) What are requirements for Digital Twin Platforms?
4) One viable system architecture
5) What is the implementation status?
6) What is to come later?
7) How can you contribute?

PRESENTATION OUTLINE

🏭 ↔️ 👥
DTaaS

1) **What is inside a Digital Twin?**
2) Who are Users?
3) What are requirements for Digital Twin Platforms?
4) One viable system architecture
5) What is the implementation status?
6) What is to come later?
7) How can you contribute?

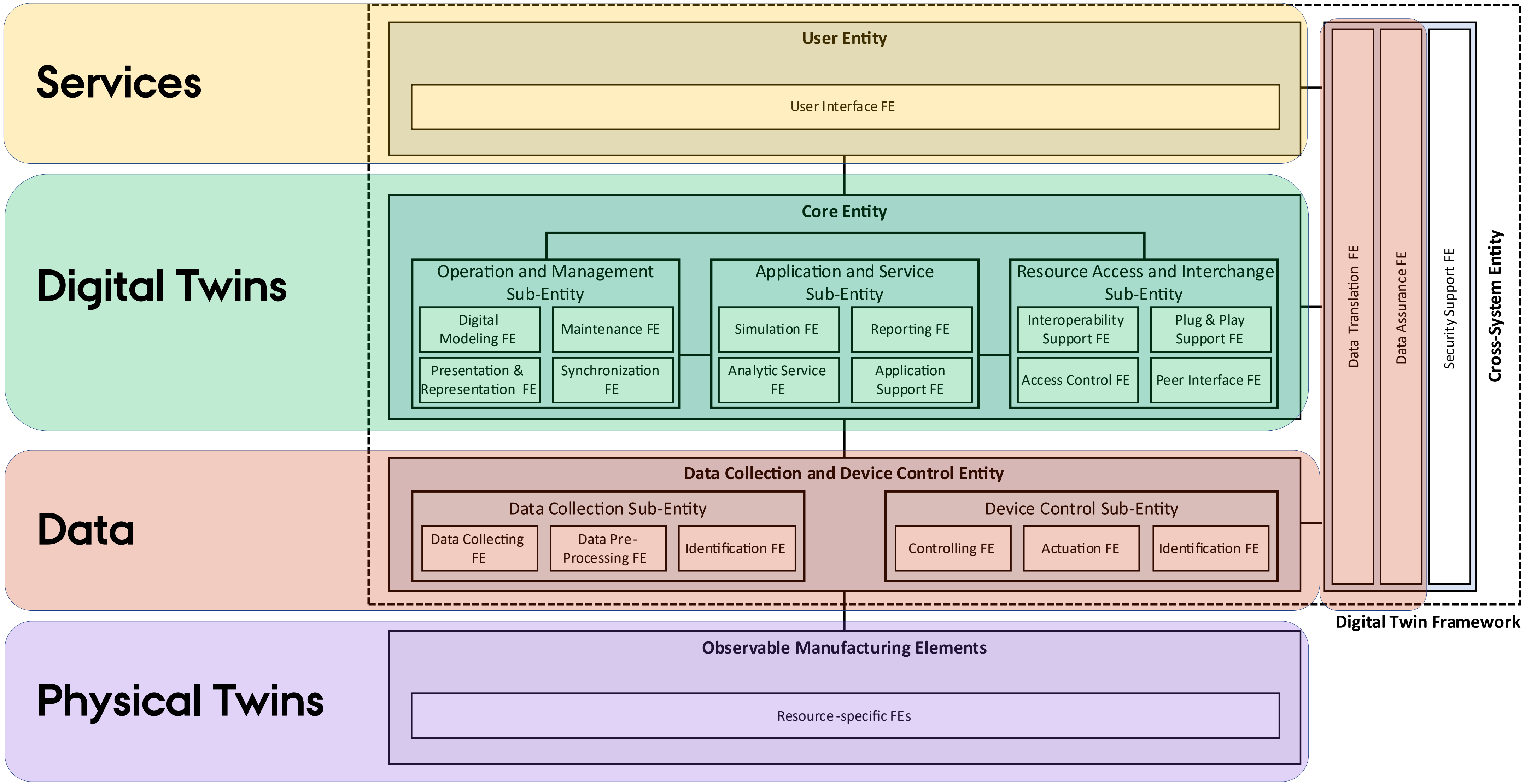
What is a Digital Twin?



- ✓ Digital representation of a real-world entity which we call the physical twin.
- ✓ Connected in real-time with physical twin.
- ✓ Offers its stakeholders a range of services

ISO 23247: DIGITAL TWIN FRAMEWORK FOR MANUFACTURING

DTaaS



Services

User Entity

User Interface FE

Digital Twins

Core Entity

Operation and Management Sub-Entity

Digital Modeling FE

Maintenance FE

Presentation & Representation FE

Synchronization FE

Application and Service Sub-Entity

Simulation FE

Reporting FE

Analytic Service FE

Application Support FE

Resource Access and Interchange Sub-Entity

Interoperability Support FE

Plug & Play Support FE

Access Control FE

Peer Interface FE

Data

Data Collection and Device Control Entity

Data Collection Sub-Entity

Data Collecting FE

Data Pre-Processing FE

Identification FE

Device Control Sub-Entity

Controlling FE

Actuation FE

Identification FE

Physical Twins

Observable Manufacturing Elements

Resource-specific FEs

Data Translation FE

Data Assurance FE

Security Support FE

Cross-System Entity

Digital Twin Framework

Ref: ISO 23247: Digital Twin Framework for Manufacturing, accessed: 04-Sep-2024

PRESENTATION OUTLINE

1) What is inside a Digital Twin?
2) **Who are Users?**
3) **What are requirements for Digital Twin Platforms?**
4) One viable system architecture
5) What is the implementation status?
6) What is to come later?
7) How can you contribute?

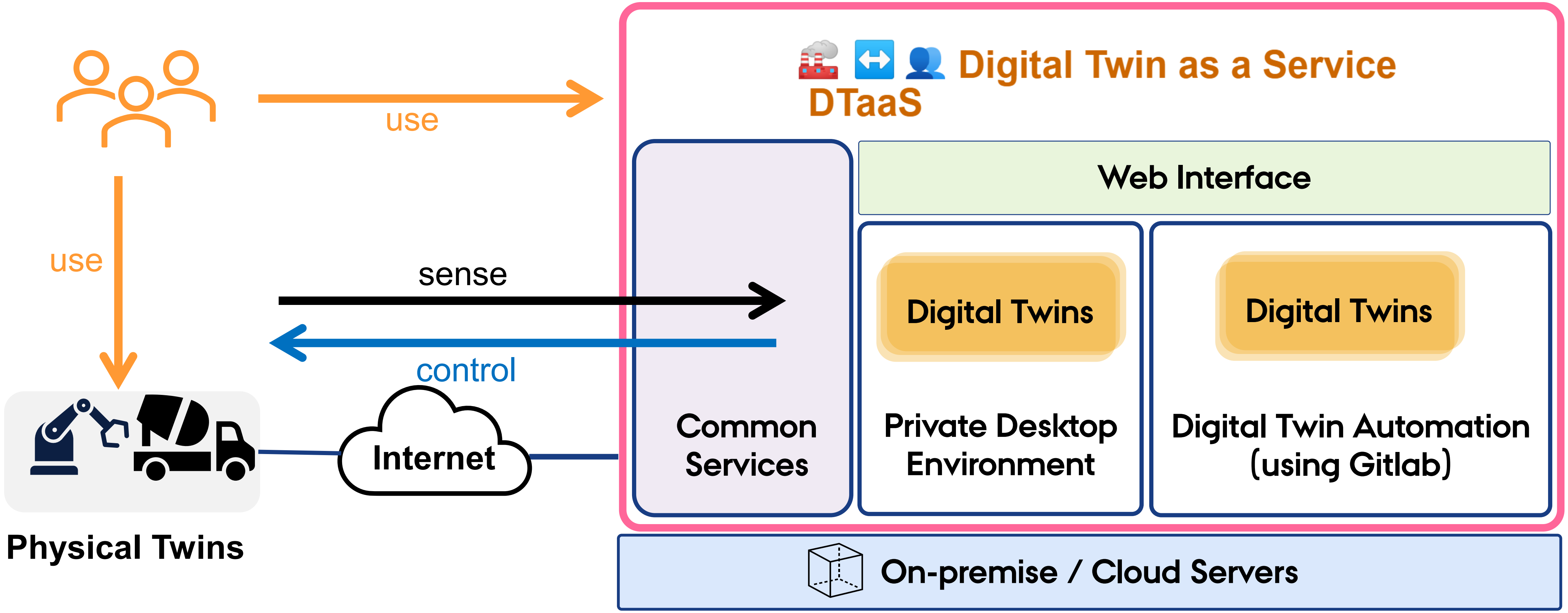
USERS AND THEIR REQUIREMENTS

User Type	Requirements
PT designer	• Access historic data from on field PTs to improve the design of future PTs.
PT manufacturing engineer	• Save data • Run DTs on DT platform. • Tweak DTs to perform what-if analysis.
Sensing PT expert	• Perform data ingestion and processing tasks. • Configure data assets for reusability.
Model producer	• *Create and maintain models for DTs.* • *Execute DTs with different models*
PT maintenance expert	• Configure DT • Use DT services
DT architect	• Compose, configure DT and required services. • Integrate DTs with internal and external services

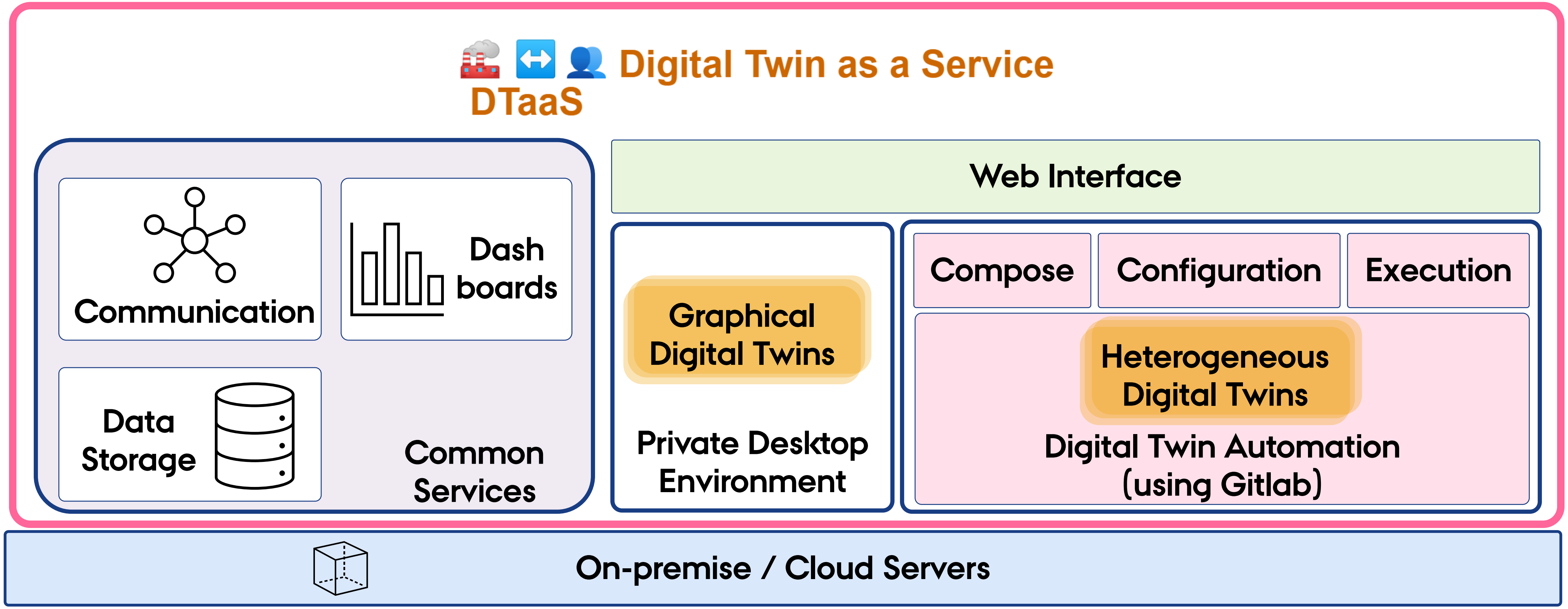
PRESENTATION OUTLINE

1) What is inside a Digital Twin?
2) Who are Users?
3) What are requirements for Digital Twin Platforms?
4) **One viable system architecture**
5) What is the implementation status?
6) What is to come later?
7) How can you contribute?

A User Centric View of DTaaS


use
use
sense
control
Internet
Physical Twins
🏭 ↔️ 👥 Digital Twin as a Service
DTaaS
Web Interface
Common Services
Digital Twins
Private Desktop Environment
Digital Twins
Digital Twin Automation (using Gitlab)
On-premise / Cloud Servers

A User Centric View of DTaaS



🏭 ↔️ 👥 Digital Twin as a Service
DTaaS

Communication

Dash boards

Data Storage

Common Services

Web Interface

Graphical Digital Twins

Private Desktop Environment

Compose

Configuration

Execution

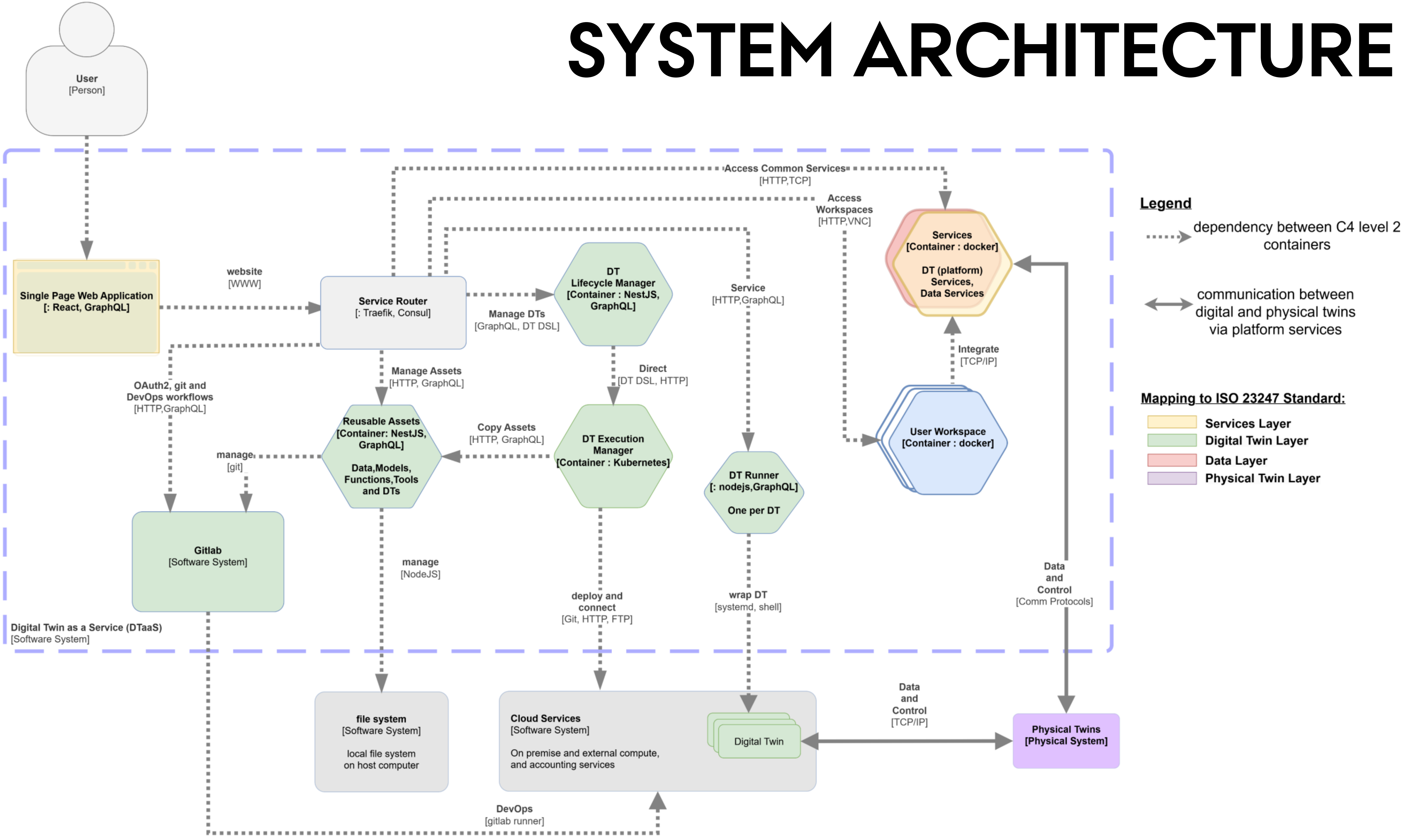
Heterogeneous Digital Twins

Digital Twin Automation (using Gitlab)

On-premise / Cloud Servers

SYSTEM ARCHITECTURE

DTaaS



User
[Person]

Single Page Web Application
[: React, GraphQL]

website
[WWW]

Service Router
[: Traefik, Consul]

Access Common Services
[HTTP,TCP]

Access
Workspaces
[HTTP,VNC]

Services
[Container : docker]

DT (platform)
Services,
Data Services

Manage DTs
[GraphQL, DT DSL]

DT
Lifecycle Manager
[Container : NestJS,
GraphQL]

Service
[HTTP,GraphQL]

Integrate
[TCP/IP]

User Workspace
[Container : docker]

OAuth2, git and
DevOps workflows
[HTTP,GraphQL]

Manage Assets
[HTTP, GraphQL]

Direct
[DT DSL, HTTP]

Reusable Assets
[Container: NestJS,
GraphQL]

Data,Models,
Functions,Tools
and DTs

Copy Assets
[HTTP, GraphQL]

DT Execution
Manager
[Container : Kubernetes]

DT Runner
[: nodejs,GraphQL]

One per DT

manage
[git]

Gitlab
[Software System]

manage
[NodeJS]

deploy and
connect
[Git, HTTP, FTP]

wrap DT
[systemd, shell]

Data
and
Control
[Comm Protocols]

Digital Twin as a Service (DTaaS)
[Software System]

file system
[Software System]

local file system
on host computer

Cloud Services
[Software System]

On premise and external compute,
and accounting services

Digital Twin

Data
and
Control
[TCP/IP]

Physical Twins
[Physical System]

DevOps
[gitlab runner]

Legend

dependency between C4 level 2 containers

communication between digital and physical twins via platform services

Mapping to ISO 23247 Standard:

🟨 Services Layer

🟩 Digital Twin Layer

🟥 Data Layer

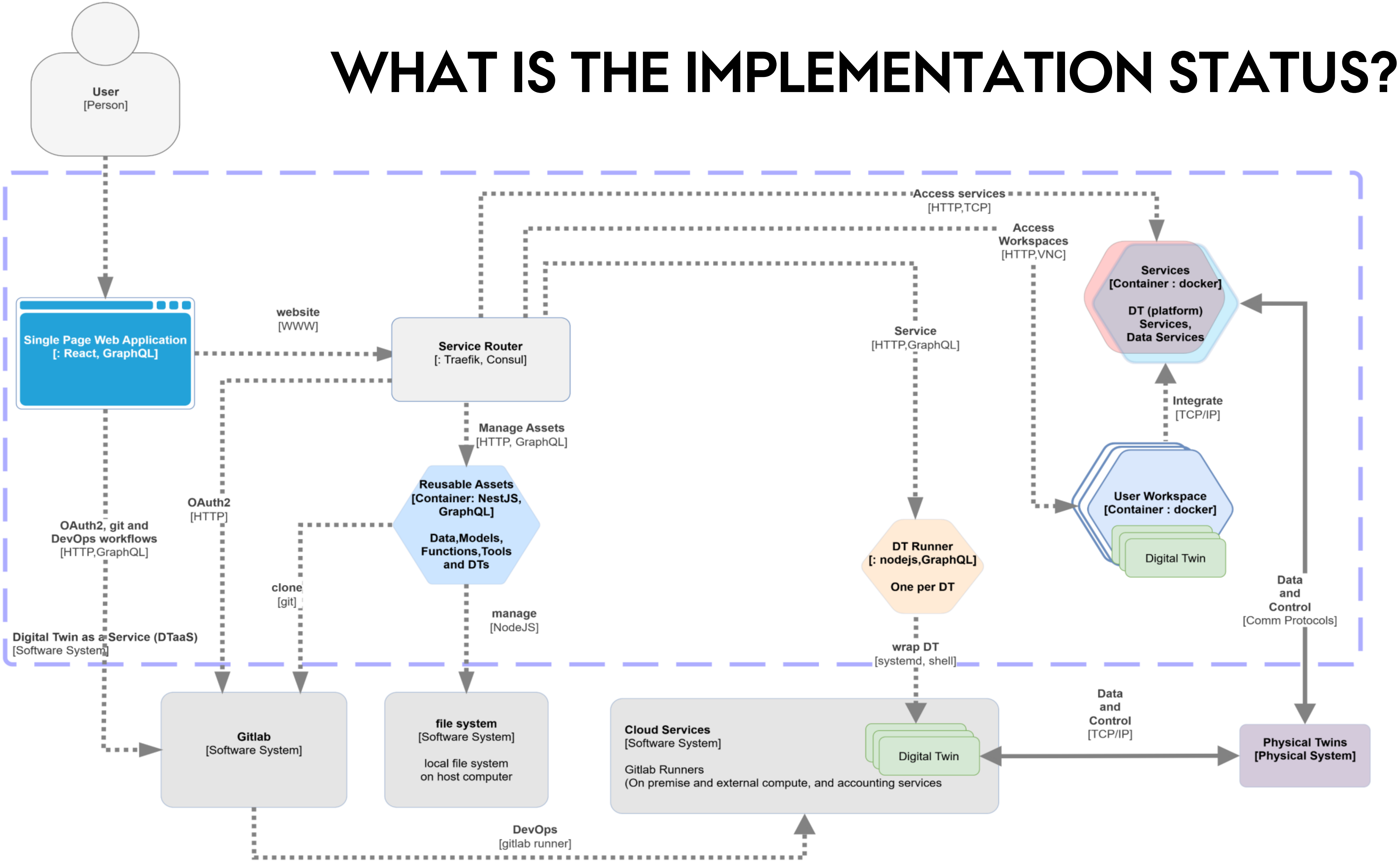
🟪 Physical Twin Layer

PRESENTATION OUTLINE

1) What is inside a Digital Twin?
2) Who are Users?
3) What are requirements for Digital Twin Platforms?
4) One viable system architecture
5) **What is the implementation status?**
6) What is to come later?
7) How can you contribute?

WHAT IS THE IMPLEMENTATION STATUS?

DTaaS



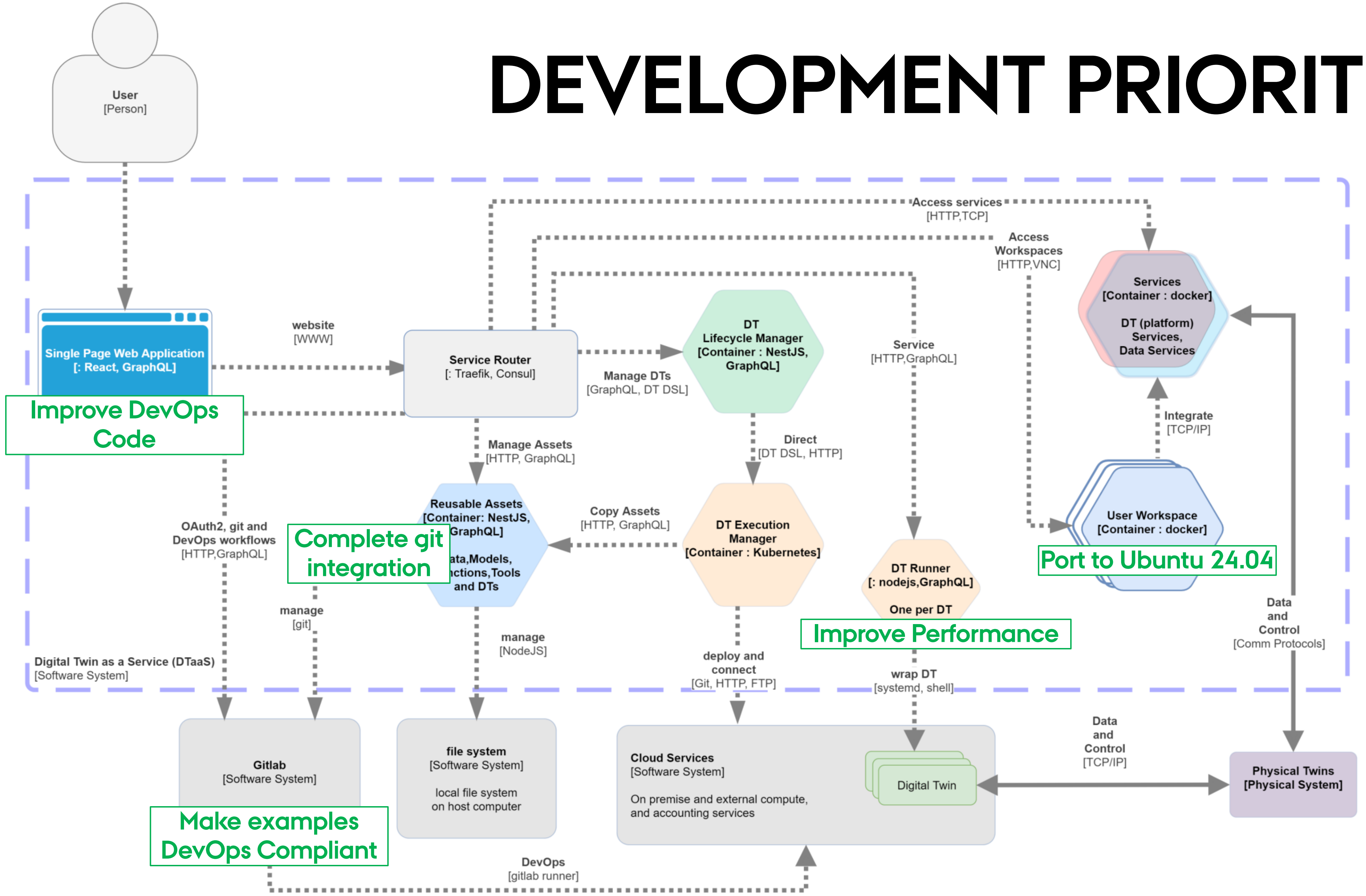
Legend

┈┈┈▶ dependency between C4 level 2 containers

◀——▶ communication between digital and physical twins via platform services

DEVELOPMENT PRIORITIES

DTaaS



User
[Person]

Single Page Web Application
[: React, GraphQL]

Improve DevOps Code

website
[WWW]

Service Router
[: Traefik, Consul]

Access services
[HTTP,TCP]

Access Workspaces
[HTTP,VNC]

Service
[HTTP,GraphQL]

DT Lifecycle Manager
[Container : NestJS, GraphQL]

Manage DTs
[GraphQL, DT DSL]

Services
[Container : docker]

DT (platform) Services,
Data Services

Integrate
[TCP/IP]

User Workspace
[Container : docker]

Port to Ubuntu 24.04

Manage Assets
[HTTP, GraphQL]

Direct
[DT DSL, HTTP]

Reusable Assets
[Container: NestJS, GraphQL]
ata,Models,
nctions,Tools
and DTs

Complete git integration

Copy Assets
[HTTP, GraphQL]

DT Execution Manager
[Container : Kubernetes]

DT Runner
[: nodejs,GraphQL]

One per DT

Improve Performance

OAuth2, git and DevOps workflows
[HTTP,GraphQL]

manage
[git]

manage
[NodeJS]

deploy and connect
[Git, HTTP, FTP]

wrap DT
[systemd, shell]

Data and Control
[Comm Protocols]

Digital Twin as a Service (DTaaS)
[Software System]

Gitlab
[Software System]

Make examples DevOps Compliant

file system
[Software System]

local file system on host computer

Cloud Services
[Software System]

On premise and external compute, and accounting services

Digital Twin

Data and Control
[TCP/IP]

Physical Twins
[Physical System]

DevOps
[gitlab runner]

Legend

⋯▶ dependency between C4 level 2 containers

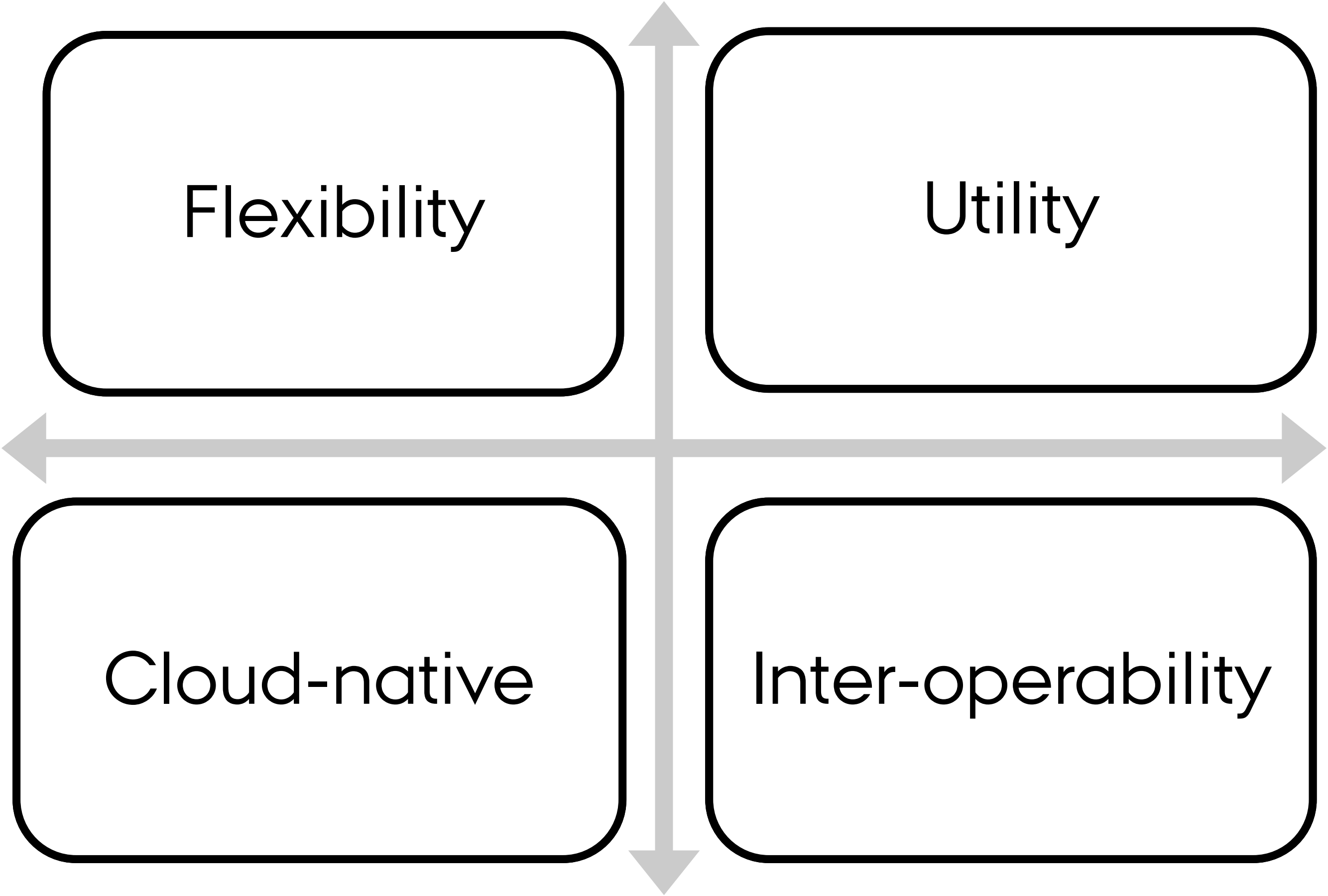
⟷ communication between digital and physical twins via platform services

PRESENTATION OUTLINE

🏭 ↔️ 👥
DTaaS

1) What is inside a Digital Twin?
2) Who are Users?
3) What are requirements for Digital Twin Platforms?
4) One viable system architecture
5) What is the implementation status?
6) **What is to come later?**
7) How can you contribute?

PRODUCT ROADMAP BY CATEGORY



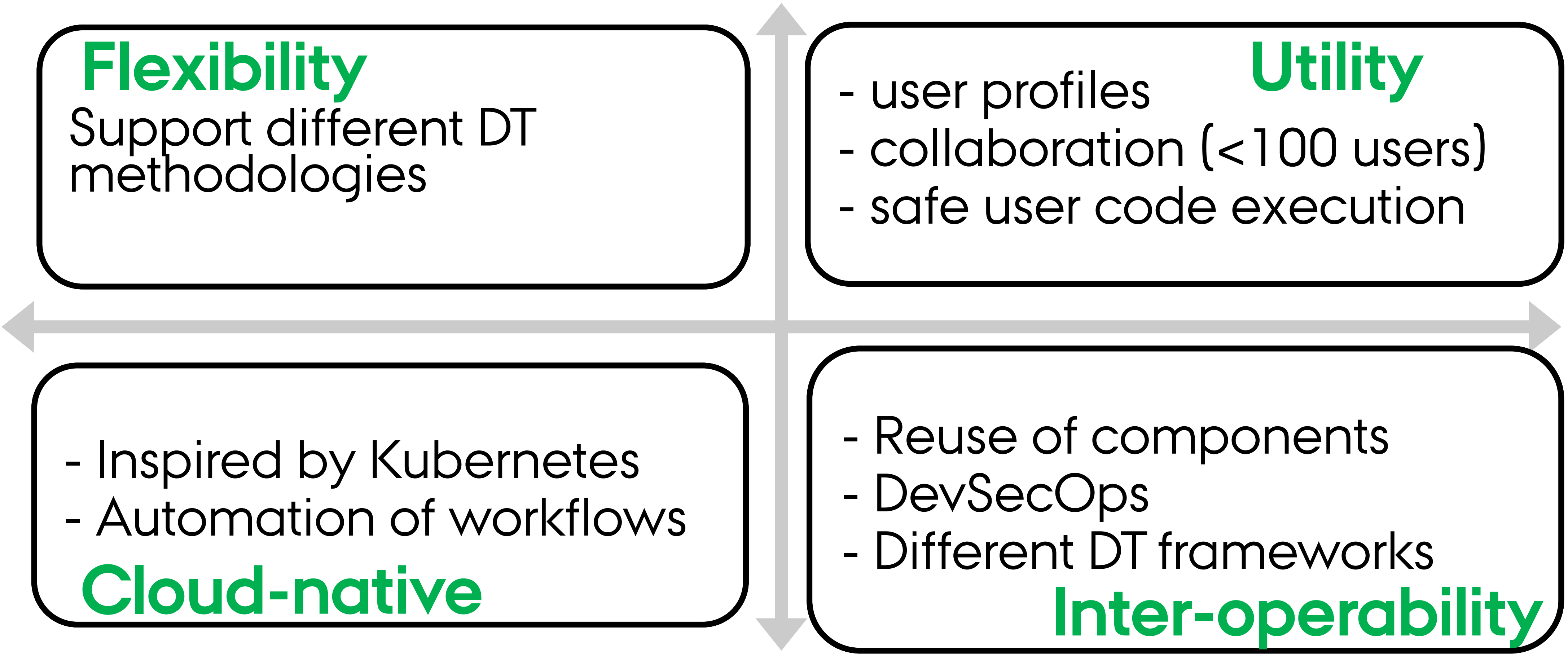
Flexibility

Utility

Cloud-native

Inter-operability

PRODUCT ROADMAP BY CATEGORY(2)



Flexibility

Support different DT methodologies

Utility

- user profiles
- collaboration (<100 users)
- safe user code execution

Cloud-native

- Inspired by Kubernetes
- Automation of workflows

Inter-operability

- Reuse of components
- DevSecOps
- Different DT frameworks

PRODUCT ROADMAP(3)

Category	Now	Next	Later
Utility	Workspace	Collaboration between users	Safe User Code Execution
Inter-operability	DevOps	Thingsboard	Eclipse BaSyX
Flexibility	Diverse Digital Twins	Different Workspaces, Lifecycle Support	Traceability, What-if, Real-time reconfiguration
Cloud-native	Docker Installation	Kubernetes, Terraform	DevSecOps

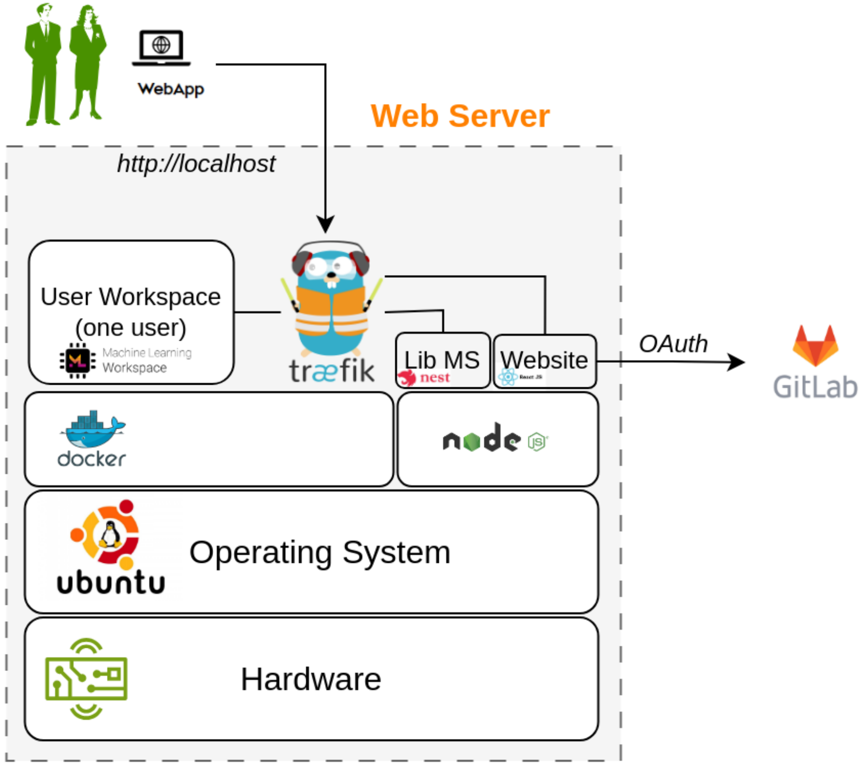
PRESENTATION OUTLINE

🏭 ↔️ 👥
DTaaS

1) What is inside a Digital Twin?
2) Who are Users?
3) What are requirements for Digital Twin Platforms?
4) One viable system architecture
5) What is the implementation status?
6) What is to come later?
7) **How can you contribute?**

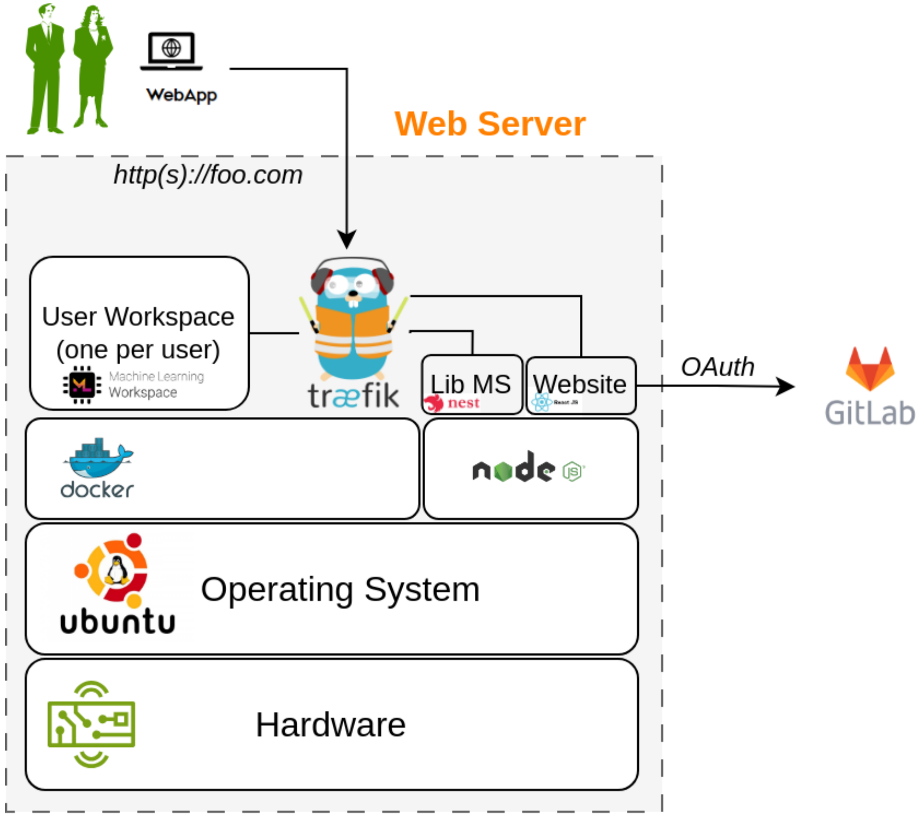
WHERE CAN YOU INSTALL THE SOFTWARE?

DTaaS



Localhost - ideal for:

- Single user
- Developement



Webserver - ideal for:

- Multiple users

TECHNOLOGY STACK

DTaaS

Web Application	Asset Management Layer (Library Microservice)	Deployment/Installation Software	Application CLI
Typescript Material UI React Redux Oauth Jest Playwright	Typescript NodeJS NestJS Jest CloudCMD	Shell Javascript Vagrant Docker	Python Pytest Click

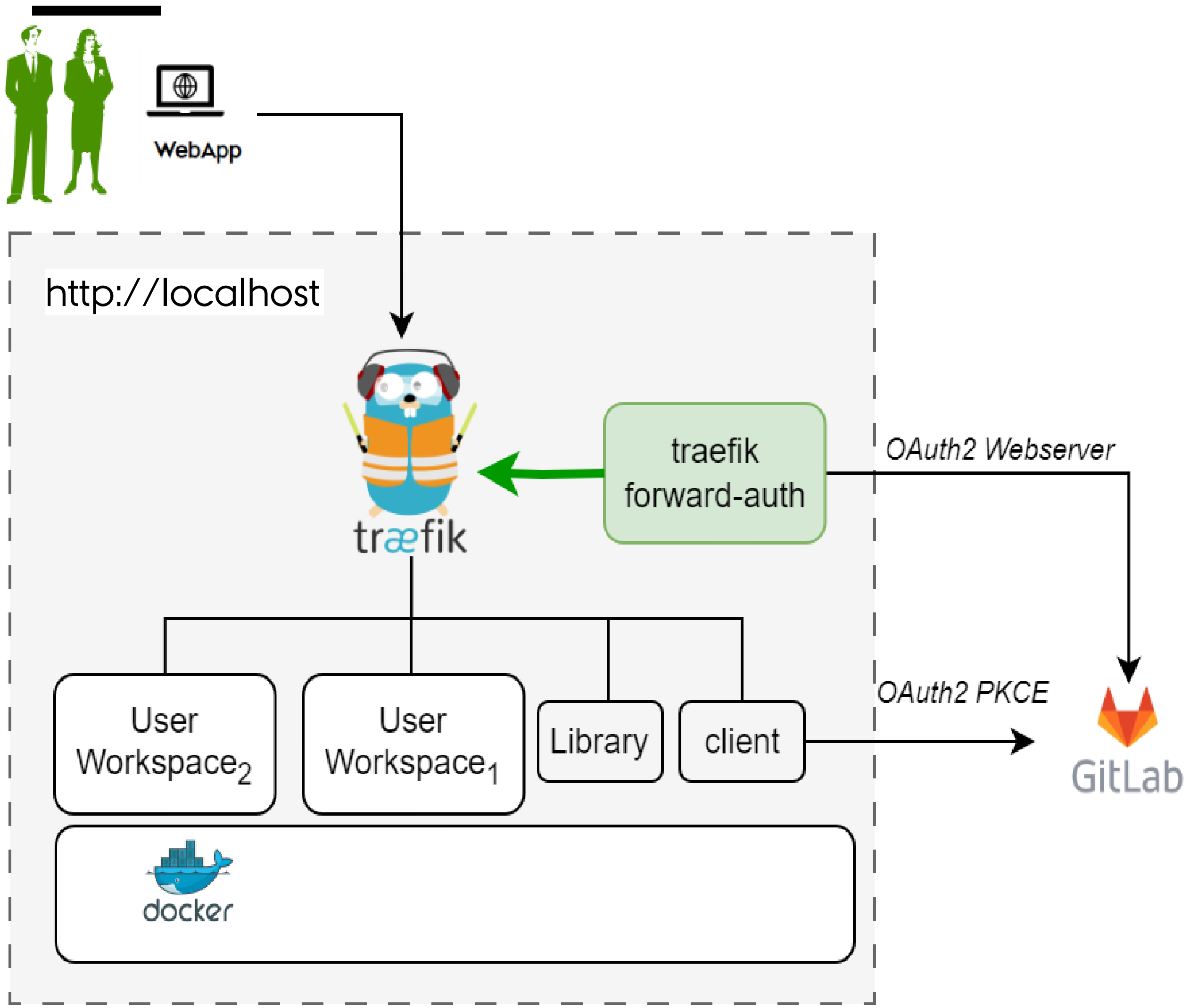
ON-BOARDING NEW DEVELOPERS

Your contributions are welcome

A few things to quickly get off the ground:

1. Read through the [documentation]() to get a birds' eye view
2. Follow [developer guidelines]()
3. Write good code
    1. follow SOLID principles
    2. Test and
    3. Maintain code quality (as measured by pre-commit hooks, codeclimate and codecov
4. Open Pull Request
5. Participate in the discussion to improve and merge the PR

DTAAS DEVELOPMENT SETUP



See docker/README.md

Remember to configure:
Docker environment variables (docker/.env)
Client (client/config/local.js)
Lib Microservice (lib/config/libms.dev.yaml)

INITIAL STEPS

🏭 ↔️ 👥 DTaaS

- See the developer information - video, docs
- Understand OAuth – concepts, client-side, Traefik forward-auth server
- Try localhost installation
- Fork the repository and
  - Setup secrects for Github Actions
  - Setup qlty, sonarcube and codecov integration (can also be done after your first contribution)
- For DevOps on DTaaS
  - See DevOps adoption in DTaaS
  - Use our hosted Gitlab but setup your own Gitlab runners
- Setup development environment
  - devContainer (see .devcontainer)

NAVIGATING THE CODE BASE

Purpose	Files
General Expectations	README.md CONTRIBUTING.md CODE_OF_CONDUCT.md LICENSE.md
Development Environment	.devcontainer/ docker/
Quality Control	.eslintrc .pre-commit-config.yaml (disfunctional) .qlty/qlty.toml .codecov.yml .pylintrc
Publish Packages	.github/

Purpose	Files
User Installation	deploy/ files/
User Management	cli/
Documentation	docs/ mkdocs-github.yml mkdocs.yml .markdownlint.yaml .mdlrc .mdl_style.rb

Navigating the React Website (client/)

Purpose	Files
Developer Help	DEVELOPER.md (please follow the advice)
Language and Package Settings	tsconfig.json package.json yarn.lock (not to be updated without a matching package version change in package.json)
Quality Control	eslint.config.mjs .prettierrc .madgerc
Static files	public/

Purpose	Files
Configuration	config/ (one template per each client environment) env.d.ts
User Installation	README.md DOCKER.md compose.client.yml .dockerignore
Source code	src/
Tests	test/ jest.config.json playwright.config.ts

Navigating the React Website (client/src)

Purpose	Files
App initialization	AppProvider.tsx index.tsx
Common functions	util/ (authorization and configuration)
Website structure	components/ page/

Purpose	Files
Different webpages	routes.tsx route/
Application state	store/
Gitlab DevOps integration	preview/ (current focus) Start with DevOps docs

To Sum Up

1. You are welcome to contribute
2. Gain experience in Digital Twins area
3. Be nice
4. Reach out to other contributors