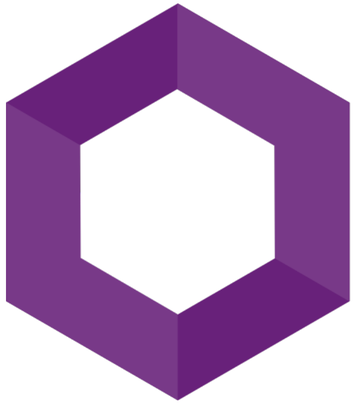
Dev>Marche

linkedin.com/in/andreabalducci/

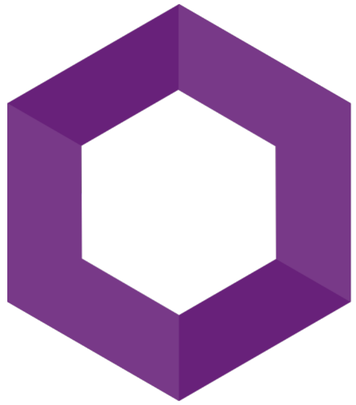


Orleans

ROBUST, SCALABLE, DISTRIBUTED APPS

GETTING STARTED

https://github.com/andreabalducci/orleans-quickstart



Orleans

Cross-platform framework for building robust, scalable distributed applications

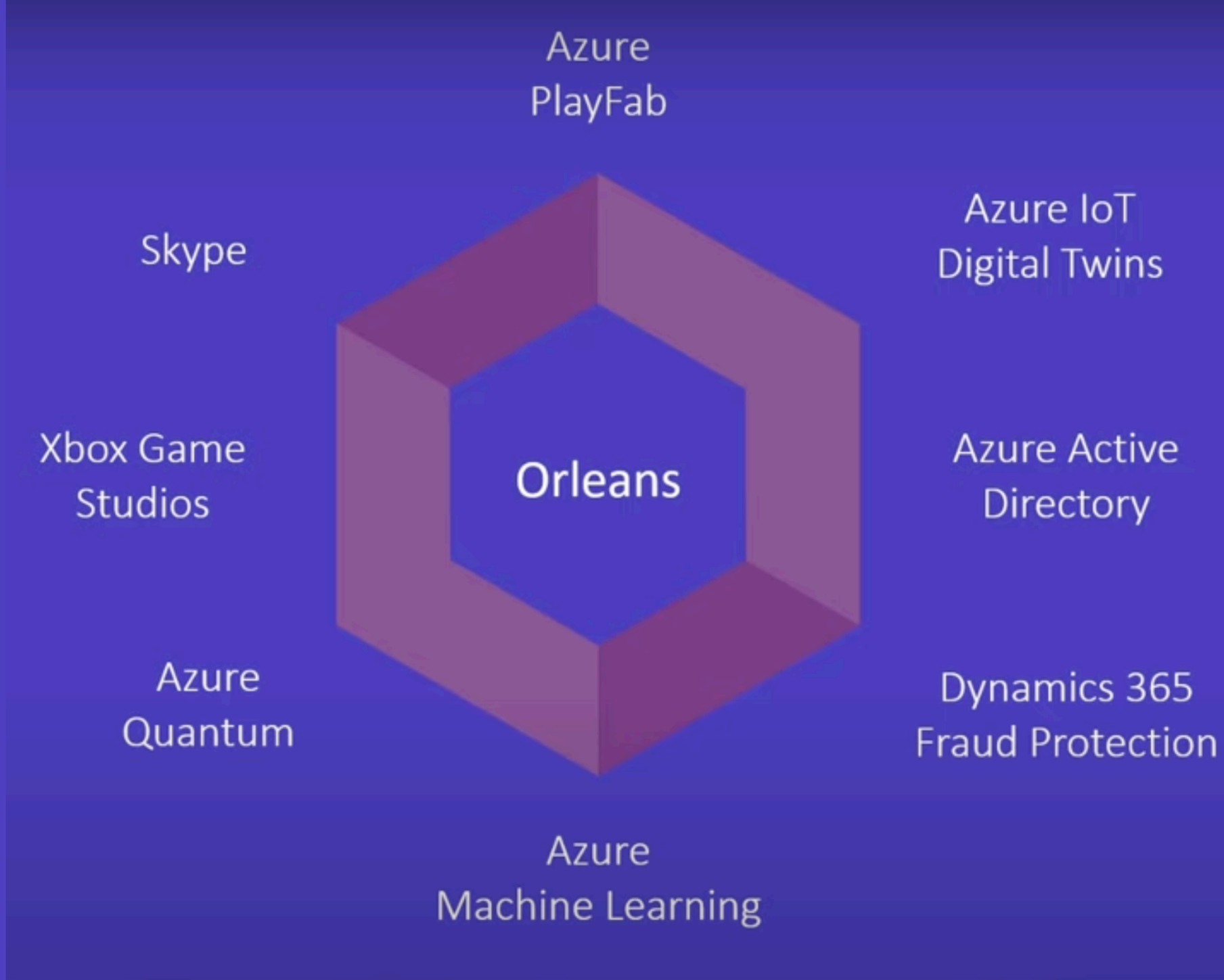
https://github.com/dotnet/orleans

FRAMEWORK

One of the primary design objectives is to simplify the complexities of distributed application development by providing a common set of patterns and APIs

FRAMEWORK

Developers familiar with single-server application development can easily transition to building resilient, scalable cloud-native services and other distributed applications using Orleans


Azure
PlayFab
Skype
Azure IoT
Digital Twins
Xbox Game
Studios
Orleans
Azure Active
Directory
Azure
Quantum
Dynamics 365
Fraud Protection
Azure
Machine Learning

Azure PlayFab

Multiplayer Server Hosting

Elastic scale dedicated multiplayer game servers

Digital Economy

Process payments, manage virtual currency, sell catalog items, and guard against fraud

Telemetry Processing

Programmatic event stream ingestion & processing

KNOWLEDGE
& WORK HUB

IOT

SCADA

WMS

MES

Choose an Orleans version

Orleans 7.0 | Orleans 3.x

VIRTUAL ACTOR MODEL

Actors are purely logical entities that *always* exist, virtually. An actor cannot be explicitly created nor destroyed, and its virtual existence is unaffected by the failure of a server that executes it.

Grain

Grains are lightweight, entities comprising user-defined identity, behavior, and state. Grains implicitly partition application states, enabling automatic scalability and simplifying recovery from failures. The grain state is kept in memory while the grain is active, leading to lower latency and less load on data stores.

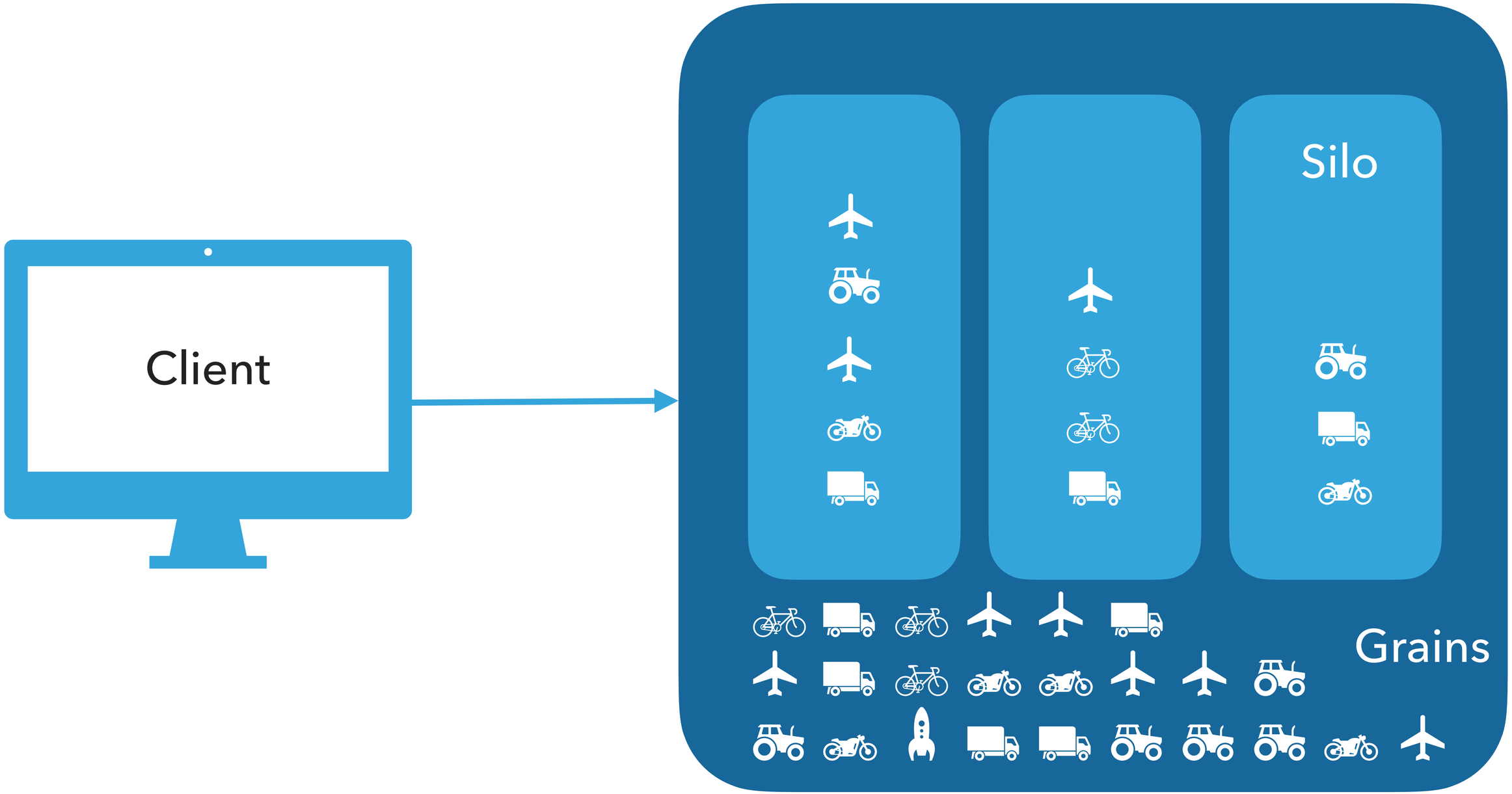
Azure PlayFab Multiplayer Servers

- Control plane for hundreds of thousands of VMs running game servers across many regions to optimize player experience
- Ensures that a sufficient pool of VMs is always ready to handle new game sessions for each game configuration
- Multiple control plane clusters for resiliency
- Grains represent tenants, titles, deployments, pools, allocators, monitors, individual VMs
- VMs send status heartbeats to control plane, control plane sends instructions to VMs
- Co-host ASP.NET Core and Orleans for simplicity and improved performance & reliability, using .NET Generic Host

Silo

A silo hosts one or more grains.

Typically, a group of silos runs as a cluster for scalability and fault tolerance. When run as a cluster, silos coordinate with each other to distribute work and detect and recover from failures.
The runtime enables grains hosted in the cluster to communicate with each other as if they are within a single process.


Client
Silo
Grains

Dashboard

Provides some simple metrics and insights into what is happening inside your Orleans application. The dashboard is simple to set up and is designed as a convenience for developers to use whilst building Orleans applications. It is not designed as a replacement for a production monitoring system.

GrainService

Grain Services exist to support cases where responsibility for servicing grains should be distributed around the Orleans cluster. Grain Services are configured on silos and are initialized when the silo starts, before the silo completes initialization. They are not collected when idle and instead have lifetimes which extend for the lifetime of the silo itself.

Stateless Worker

Requests made to stateless worker grains are always executed locally, that is on the same silo where the request originated, either made by a grain running on the silo or received by the silo's client gateway. Hence, calls to stateless worker grains from other grains or client gateways never incur a remote message.

Request Context

Metadata and other information can be passed with a series of requests using the request context. Request context can be used for holding distributed tracing information or any other user-defined values.

Reminders

Reminders are a durable scheduling mechanism for grains. They can be used to ensure that some action is completed at a future point even if the grain isn't currently activated at that time.

Timers

Timers are the non-durable counterpart to reminders and can be used for high-frequency events, which don't require reliability.

Persistence

Orleans provides a simple persistence model that ensures the state is available before processing a request, and that its consistency is maintained. Grains can have multiple named persistent data objects.

Distributed ACID Transactions

Multiple grains can participate in ACID transactions together regardless of where their state is ultimately stored. Transactions in Orleans are distributed and decentralized (there is no central transaction manager or transaction coordinator) and have serializable isolation.

Testing

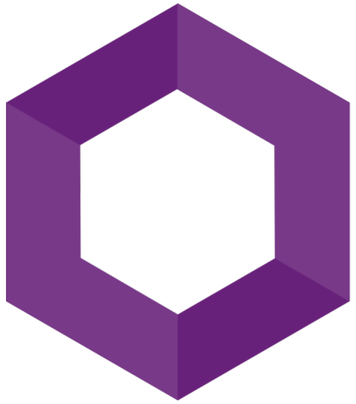
The Microsoft.Orleans.TestingHost NuGet package contains TestCluster which can be used to create an in-memory cluster, comprised of two silos by default, which can be used to test grains.

Streaming

A stream is a logical entity that always exists and can never fail.

Streams allow the generation of data to be decoupled from its processing, both in time and space

Dev>Marche

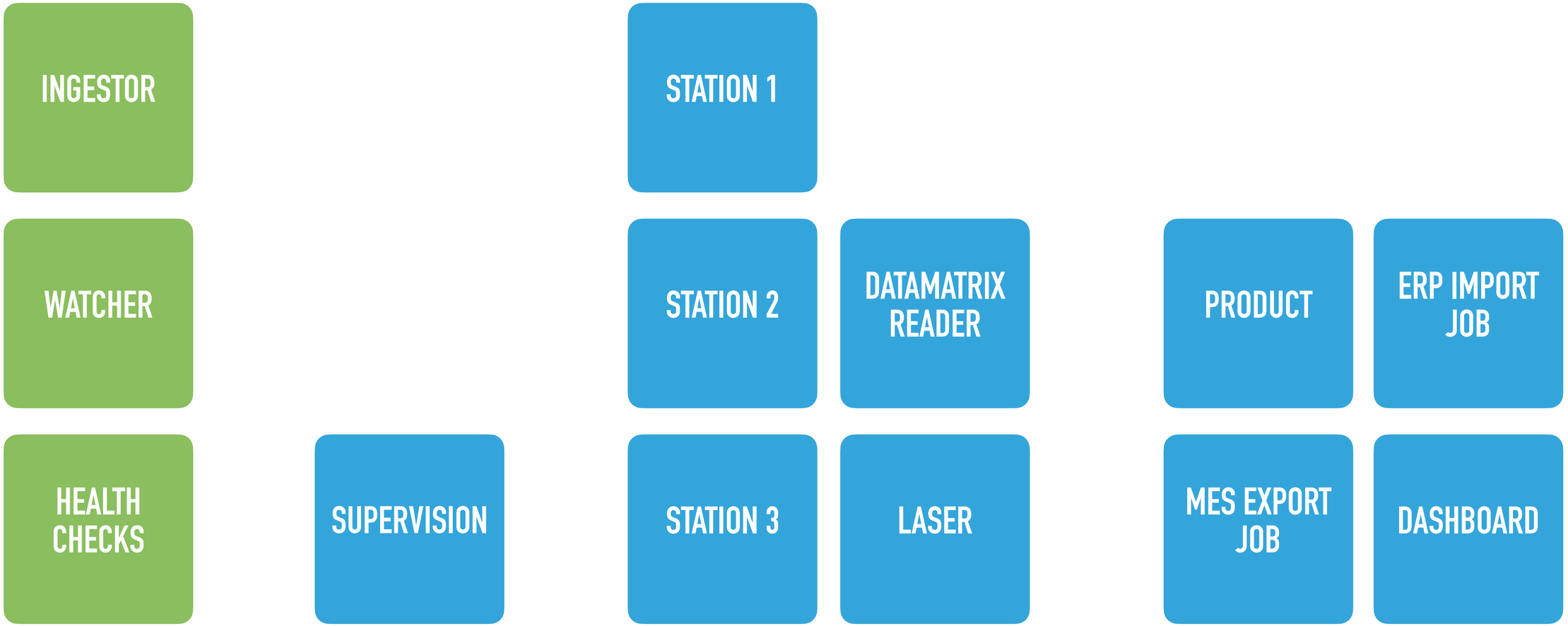


Orleans

CASE STUDY

SUPERVISORY CONTROL AND DATA ACQUISITION

ASSEMBLY LINE GRAINS (SIMPLIFIED)



ARCHITECTURAL OVERVIEW