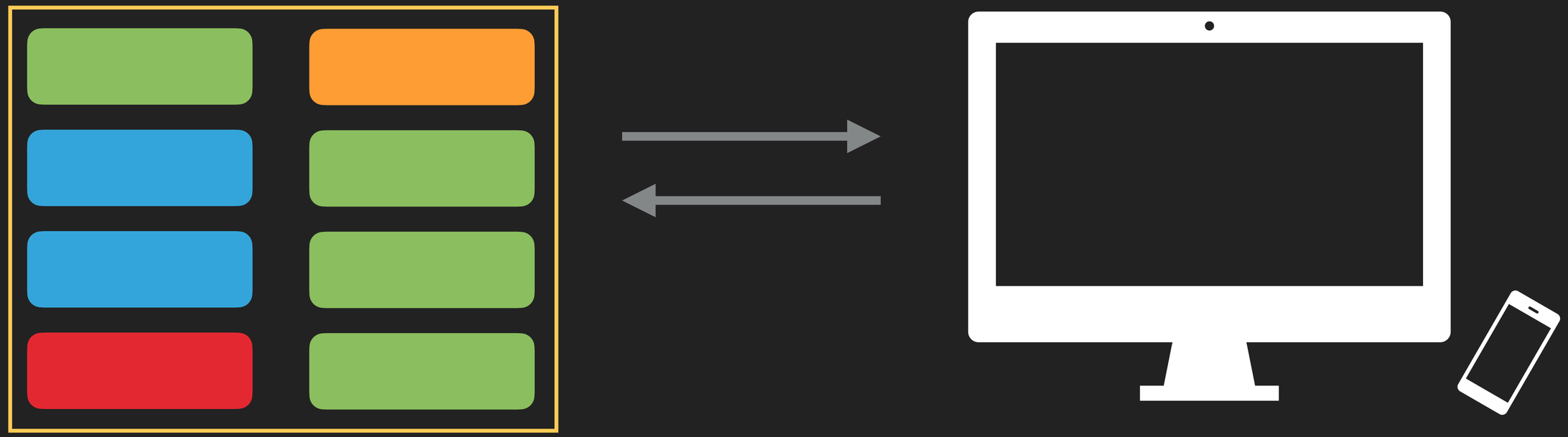


CASE STUDY

WEB DEVELOPMENT

OVERVIEW

JAPPWARE

17 PROJECTS

15 CLIENTS

537 SERVICES

3308 TESTS

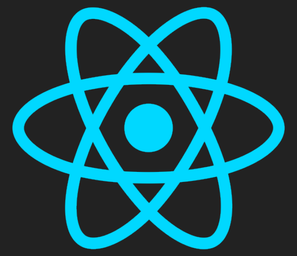
811 ENDPOINTS

5 TEAMS

FRAMEWORKS & TECHNOLOGIES

JAPPWARE

Front-End



React.JS

Angular

Vue.JS

Rx.JS

Back-End

Common top frameworks:

...{ Many others }...

Cloud services on top providers:

Enterprise languages:

PLATFORMS

JAPPWARE

Front-End

All Platforms

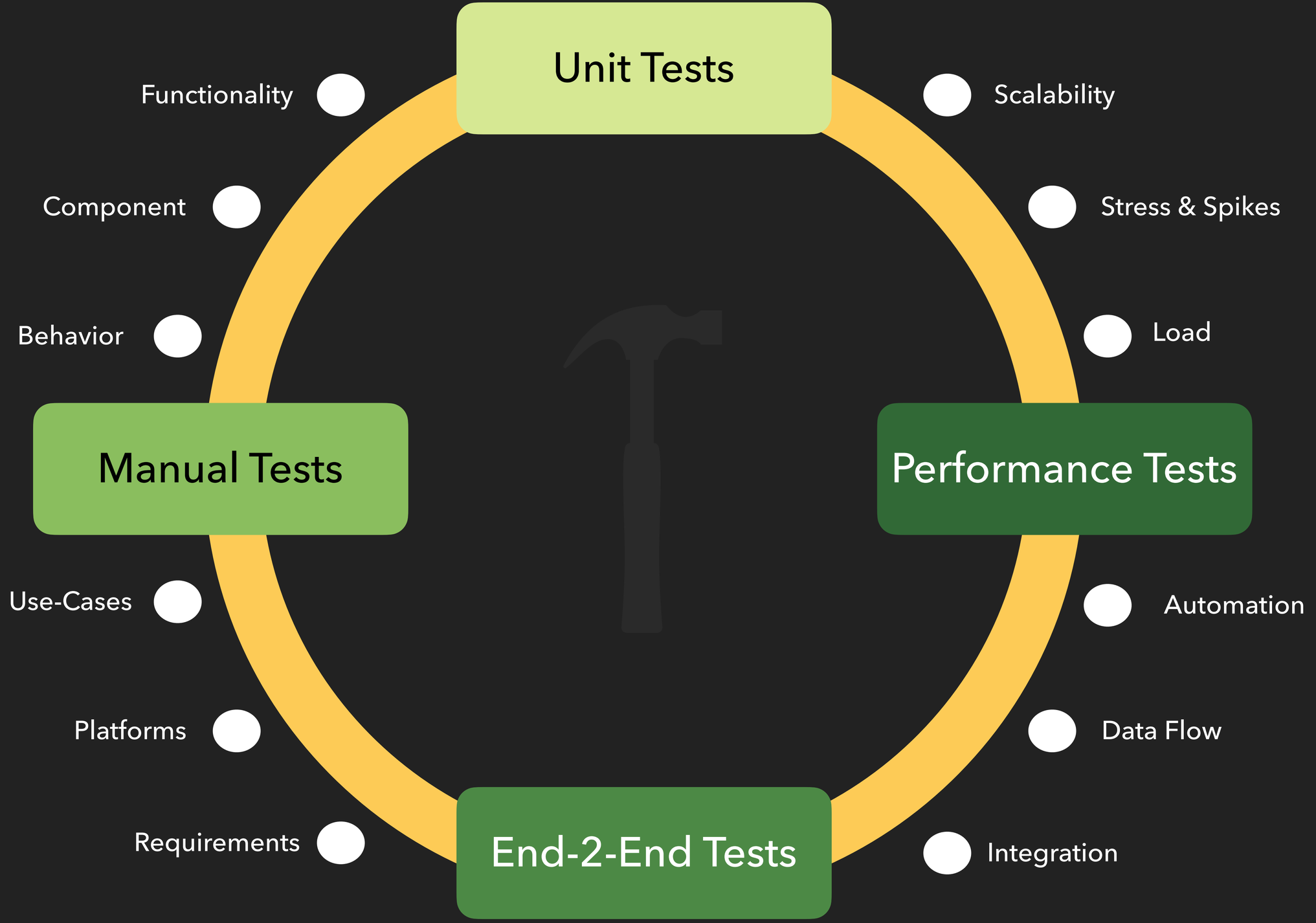
Back-End

Cross-Platform solutions

No dependencies - Better maintainability

TESTS

JAPPWARE



DEPLOYMENT

JAPPWARE

- ✓ Blue-green & Canary
- ✓ Automated up- and down- scale
- ✓ Automated infrastructure creation
- ✓ Containerisation and orchestration
- ✓ Infrastructure as a code
- ✓ CI/CD Automation

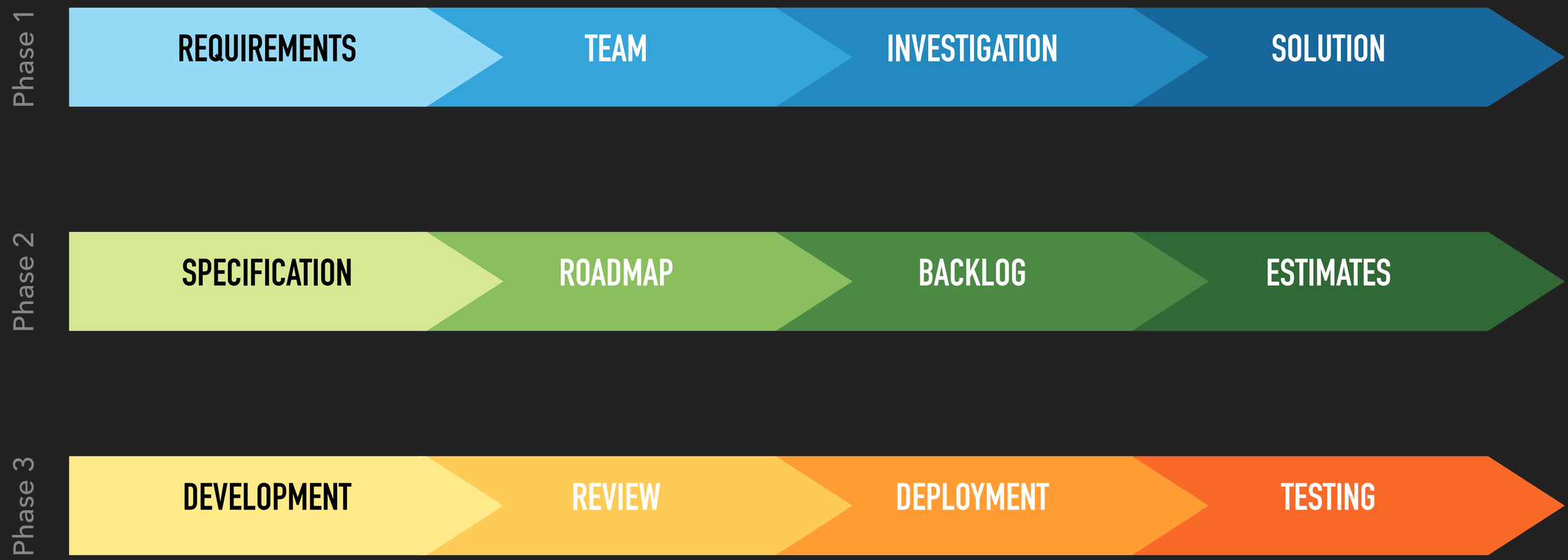
SECURITY

JAPPWARE

- ✓ Complete by OWASP top requirements
- ✓ Secured communication
- ✓ Privacy policies and Private Data protection
- ✓ DDoS attack protection
- ✓ Fraud detection

PROCESS

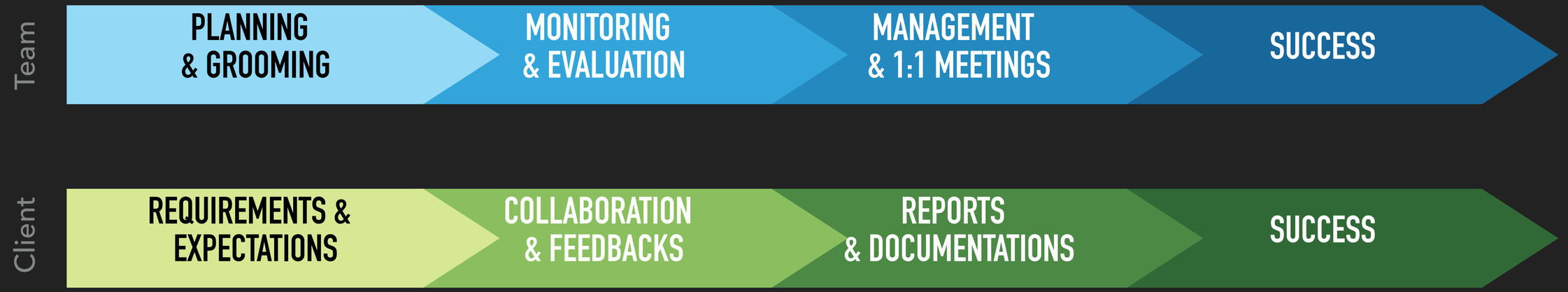
JAPPWARE



KPI

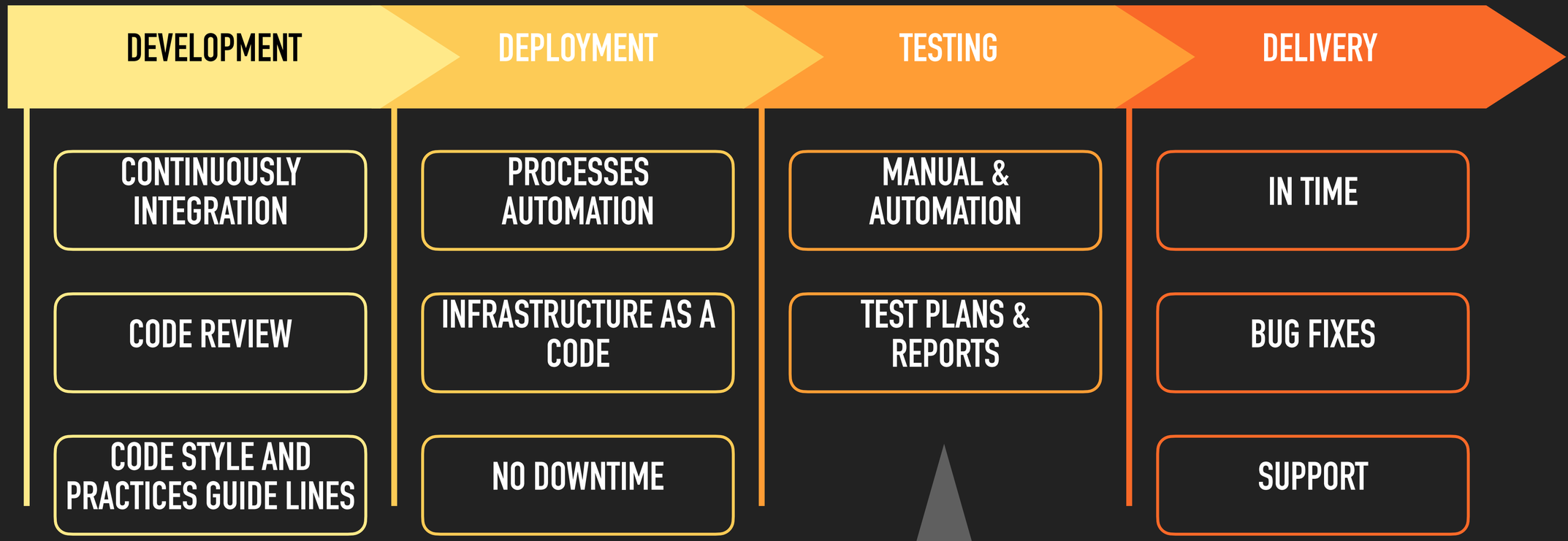
JAPPWARE

- ✓ Sprint planing & grooming meetings
- ✓ Spring acceptance meetings
- ✓ Staging delivery and test report
- ✓ Team velocity and capacity evaluation



LIFE CYCLE

JAPPWARE



Unit, Integration, Self, End-to-End & Penetration Tests

CASE #1

JAPPWARE

ON-PREMISES

ON-PREMISES

JAPPWARE

Task

Build a fault-tolerant and highly available system that provides distributed data computation on the backend for a stream of data and custom front-end application to access runtime and historical statistic with minimal latency (less than 500 ms).

Overview

Microservices solution that receives a stream of the data, including CRUD REST API, that computes output results in a distributed way to provide fast realtime vision using Lambda Architecture. User management, rights and attributes to manage data access levels and flexible Front-End to work through mobile and desktop platforms. Custom design and components implementation, CI/CD and Blue-Green deployment strategy.

Challenges

- ✓ Microservices management
- ✓ Secured internal and external communication
- ✓ CI & CD implementation
- ✓ Lambda Architecture
- ✓ Distributed Job execution

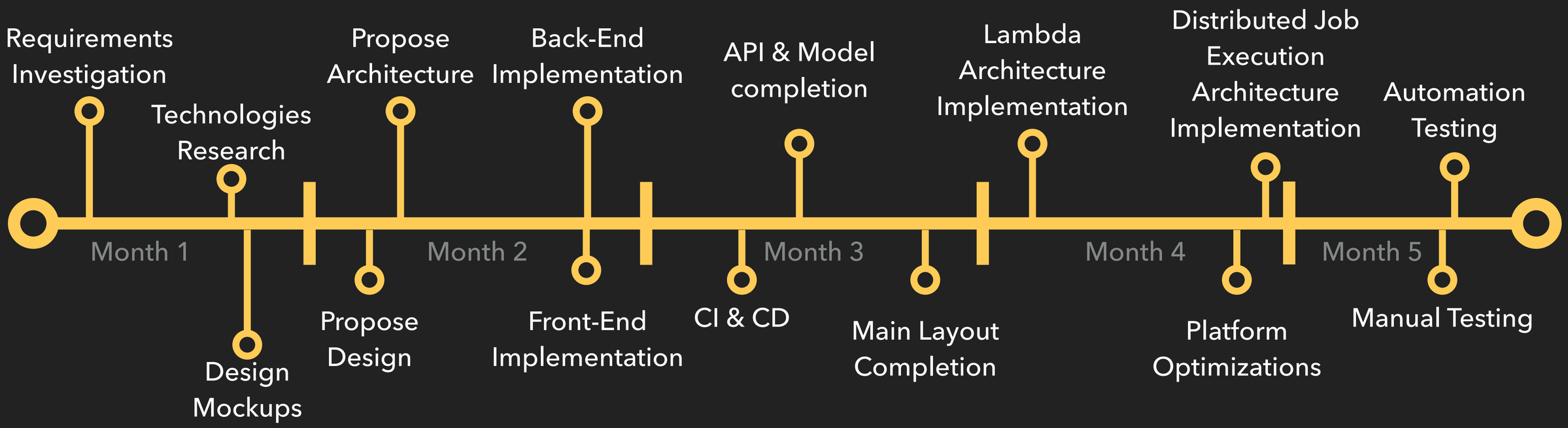
ON-PREMISES

JAPPWARE

Goals

- ✓ Build a flexible and easy deployed platform using AWS cloud provider
- ✓ Distributed Job processing using fault-tolerant Executor (Apache Spark, Apache Storm)
- ✓ Responsive Front-End implementation
- ✓ Performance

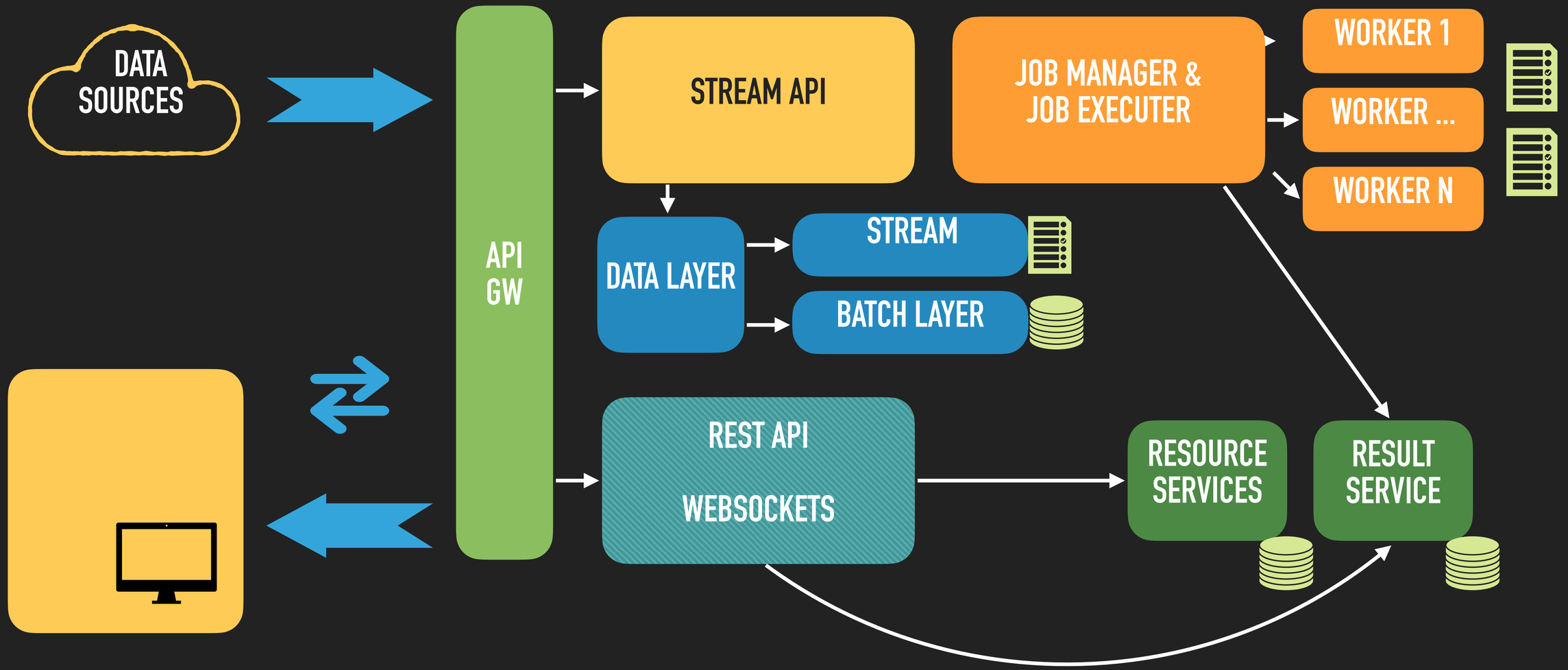
RoadMap



ON-PREMISES

JAPPWARE

Architecture



ON-PREMISES

JAPPWARE

Actions

- ✓ Use Apache Spark for Stream and Batch processing
- ✓ Implement Data layer that depends from configuration splits income traffic for Stream and Batch layer
- ✓ Use Angular and a main Front-End framework
- ✓ Used Kubernetes for Microservices & worker orchestration
- ✓ Used Project Reactor for Stream API implementation
- ✓ Used WebSockets for Push notifications

Results

- ✓ With Lambda architecture we reached low-latency API that provides realtime (but not exact) results
- ✓ Having WebSocket connection our users received immediate data updates after each task completion
- ✓ No down-time deployment & replacement was reached using Blue-Green strategy and containerisation
- ✓ REST API was used for common CRUD operations and didn’t affect the main Stream API

CASE #2

JAPPWARE

CLOUD-BASED

CLOUD-BASED

JAPPWARE

Task

3rd party integration hub. Implement the system where users may build a chain, including expressions, of different 3rd party services and create the pipeline of different operations based on events received from those services. Manage this pipeline through responsive front-end and receive a detailed log of activities behind the scene.

Overview

Distributed computation engine based or different tasks and sub-tasks with support of graceful shutdown mechanism. Flexible and robust Front-End application that allows users to manage their operations pipelines and trigger different actions using all available 3rd party integration.

Challenges

- ✓ Asynchronous task execution and result merging
- ✓ Graceful shutdown of each job execution
- ✓ Front-End pipeline builder and Back-End expression engine
- ✓ Continuous reading from 3rd parties and webhook controllers for 3rd party events

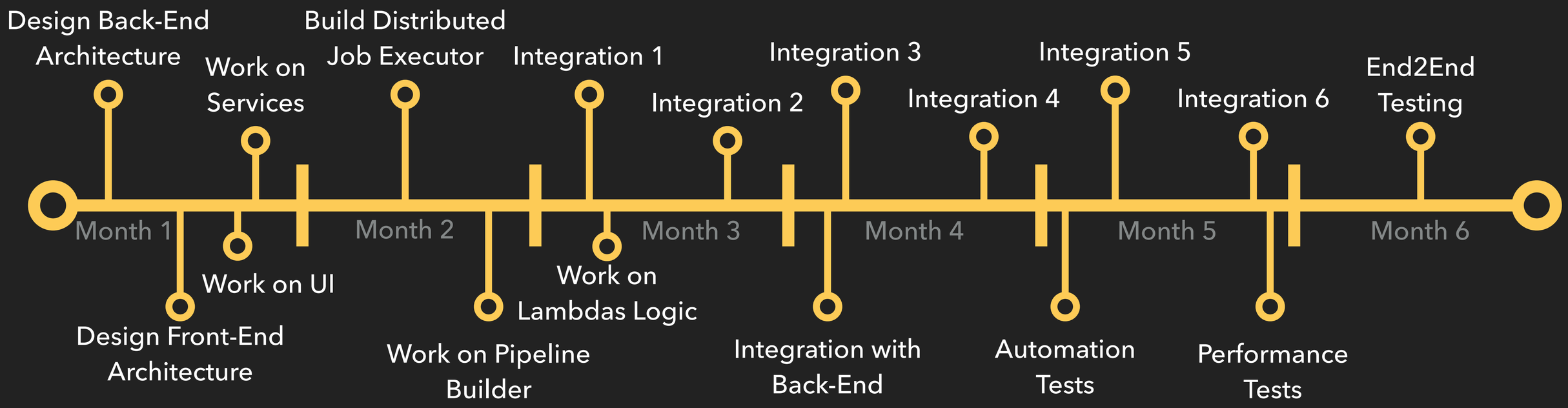
CLOUD-BASED

JAPPWARE

Goals

- ✓ Distributed Job Execution and Async task parallelization
- ✓ Component state manager
- ✓ 3rd party authentication and job execution with a graceful shutdown mechanism
- ✓ Flexible & Custom front-end implementation with the strong user experience of tasks pipeline builder

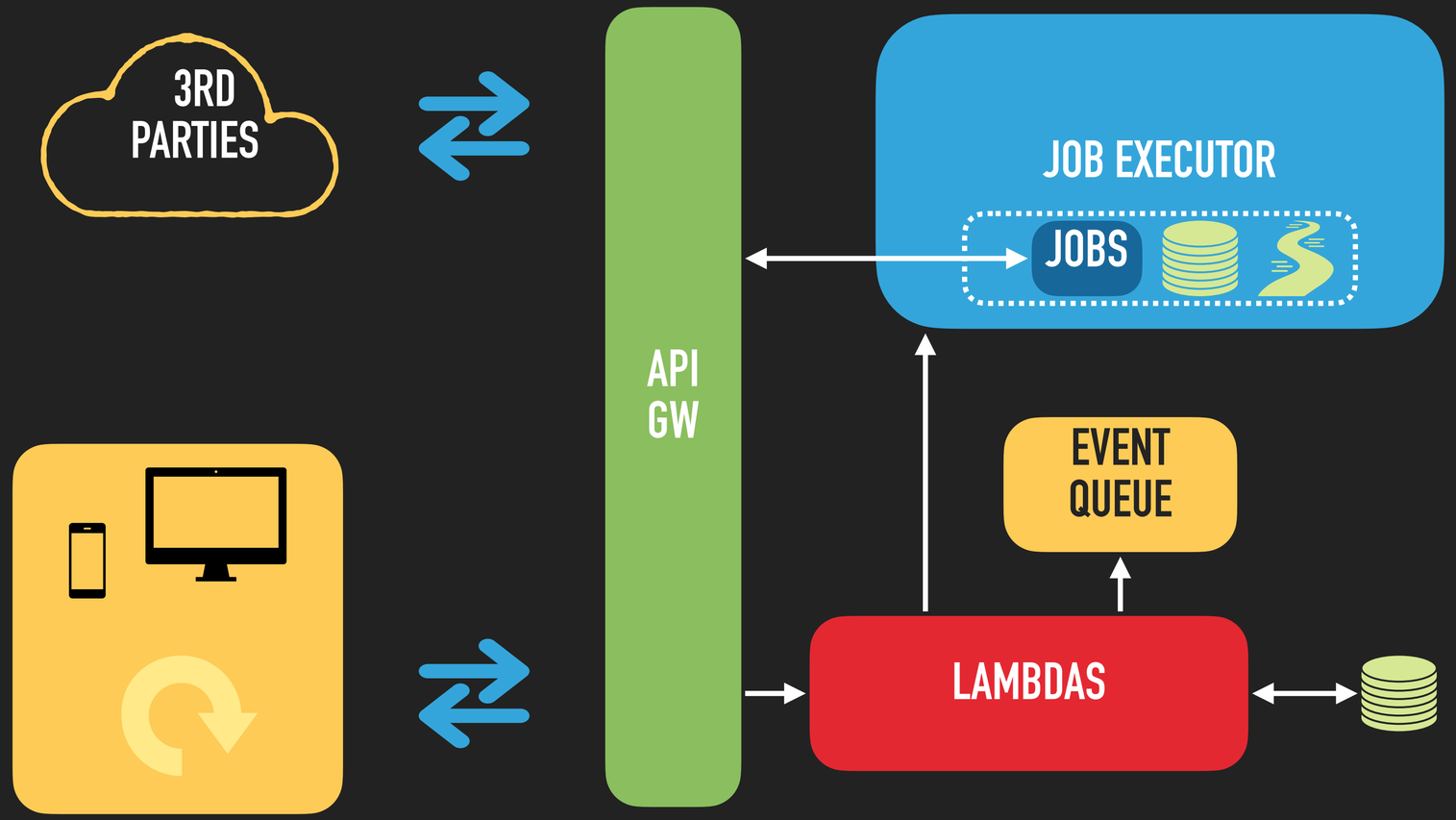
RoadMap



CLOUD-BASED

JAPPWARE

Architecture



aws

CLOUD-BASED

JAPPWARE

Actions

- ✓ Implement Framework for Distributed Job Execution on AWS SWF for further extensions
- ✓ Used API Gateway and AWS Lambdas for common CRUD operations via REST API
- ✓ Event queue to trigger a sequence of Lambdas which Trigger Job Execution
- ✓ Databases and Data Streams for Job state management and 3rd party integration details

Results

- ✓ Robust and scalable solution that executes pipeline of pre-configured actions based on events and data received from 3rd parties
- ✓ Build logging, and metric monitoring solutions using Cloud-native Services
- ✓ Responsive Front-End with custom design and components, highly experienced approach of action pipeline builder
- ✓ CI/CD solution and Infrastructure As a Code configuration

MULTITENANCY

MULTITENANCY

JAPPWARE

Task

The system that supports content management through different organizations based on different groups/departments with particular users that might belong to many of them at the same time but with different attributes and access levels to the resources.

Overview

Build a Web Application that supports SSO auth approach for all users and separate content by user's groups and organizations. Manage user's attributes and roles to provide access level flexibility. Restrict access to data based on particular user configuration in the group and protect data access between organizations, groups, and users.

Challenges

- ✓ Custom SSO solution for all customer services including new development and possible future solutions
- ✓ ABAC functionality to manage attribute-based access configuration
- ✓ The secured approach of Data retrieving to avoid illegal access issues

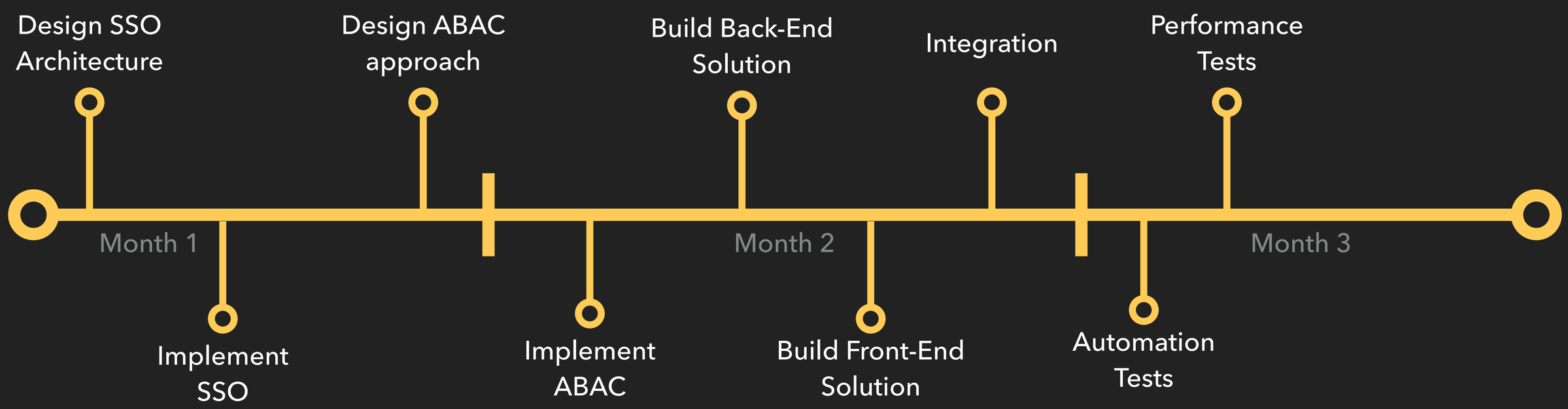
MULTITENANCY

JAPPWARE

Goals

- ✓ ABAC-based approach
- ✓ SSO implementation
- ✓ JWT auth approach
- ✓ Back-End and Front-End according to content management requirements

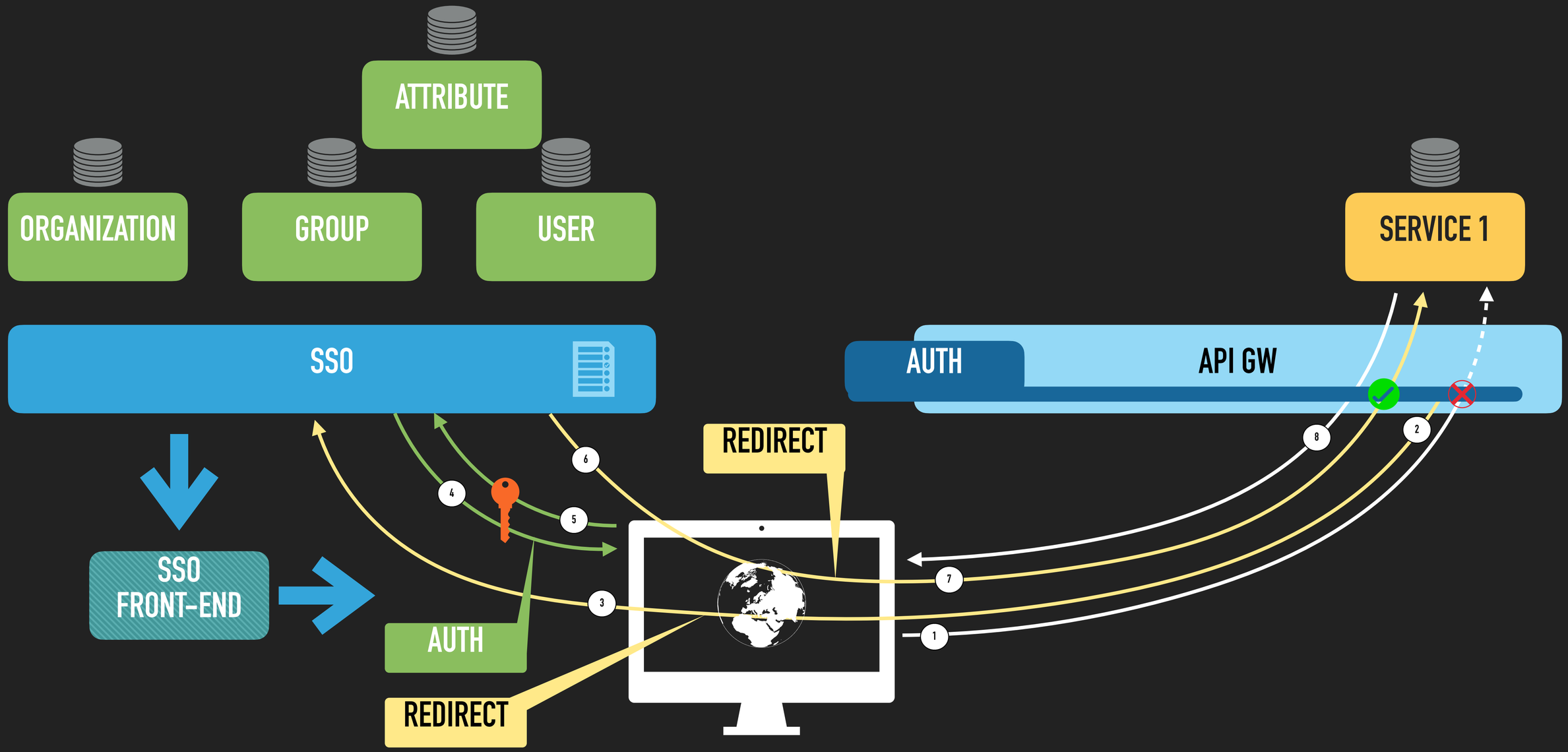
RoadMap



MULTITENANCY

JAPPWARE

Architecture



MULTITENANCY

JAPPWARE

Actions

- ✓ Implemented custom SSO server with responsive AUTH and User Management Front-End
- ✓ Used JWT approach for request identity and access verification
- ✓ Implemented API Gateway with AUTH validation and permission check functionality
- ✓ Implemented microservices architecture for future extensions and scalability

Results

- ✓ Robust SSO for all client's products
- ✓ Highly customizable tribute based approach to manage rights and access levels
- ✓ Stateless approach to identify a user
- ✓ Middleware to authorize each request

CASE #4

JAPPWARE

WEB

WEB

JAPPWARE

Task

Build an application without back-end business logic that consumes data from IoT Cloud Hub and provides device list, statistic and device management operations.

Overview

A Web Application with continuous reactive connection to the IoT Cloud Hub API that subscribed on corresponded topics to handle devices state and receive runtime statistic updates. Also, using different topics, trigger the execution of different management tasks e.g. firmware update, reboot, logs uploading

Challenges

- ✓ Complex Web Application communication architecture
- ✓ Custom design and components
- ✓ The reactive paradigm of programming
- ✓ Scalability, Availability, and Fault-Tolerance

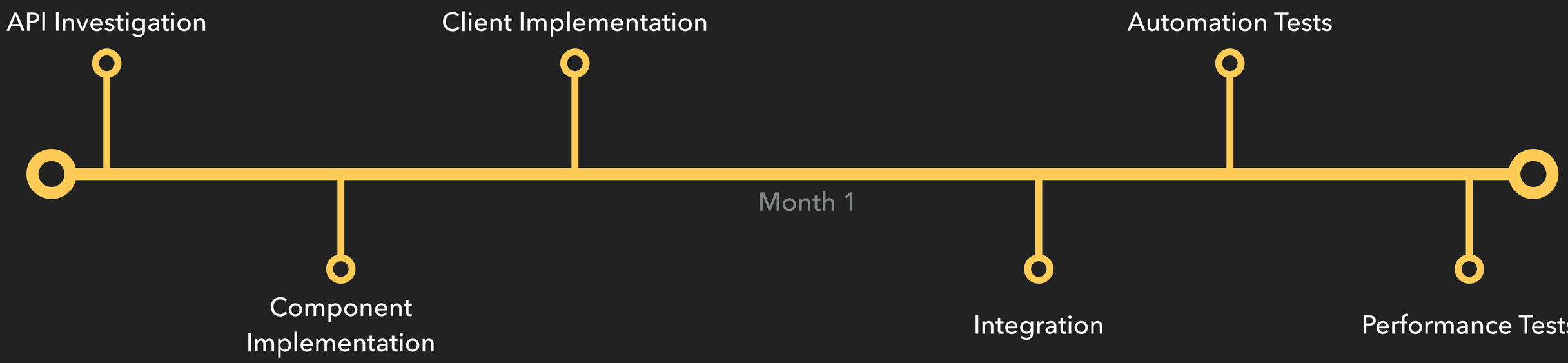
WEB

JAPPWARE

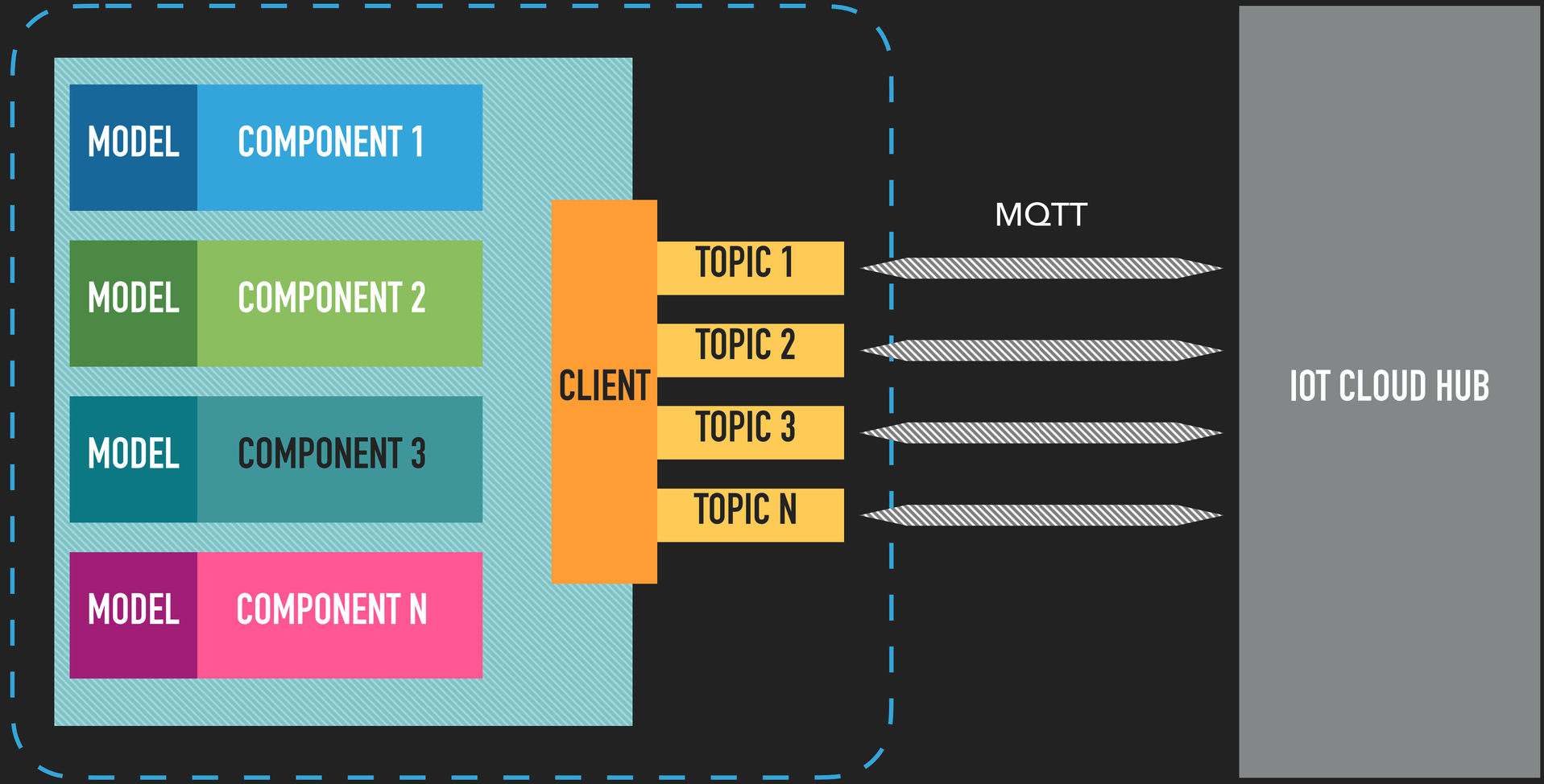
Goals

- ✓ Build reactive communication using RxJS
- ✓ Implement custom design
- ✓ Automation tests

RoadMap



Architecture



Actions

- ✓ Implemented reactive MQTT client
- ✓ Implemented custom components, models and custom design

Results

- ✓ Reactive communication channel with data flow transformation and aggregation
- ✓ Responsive and flexible Front-End with custom design and components
- ✓ Scalable solution for multiple users with support of multiple devices, stats, and simultaneous operations

CASE #5

JAPPWARE

SECURITY

SECURITY

JAPPWARE

Task

Microservices solution with secured communication between each component with a regularly changeable certificate. Secure communication over HTTP (internal and external).

Overview

The secured system with internal communication protection between all microservices independently from language or technologies used for each particular service. Attack-tolerant by OWASP top 10 Security Risks.

Challenges

- ✓ Secured internal communication
- ✓ Continuous process of certificate renewing and delivering to all service
- ✓ Automation tests using Top 10 Web Application Security Risks by OWASP

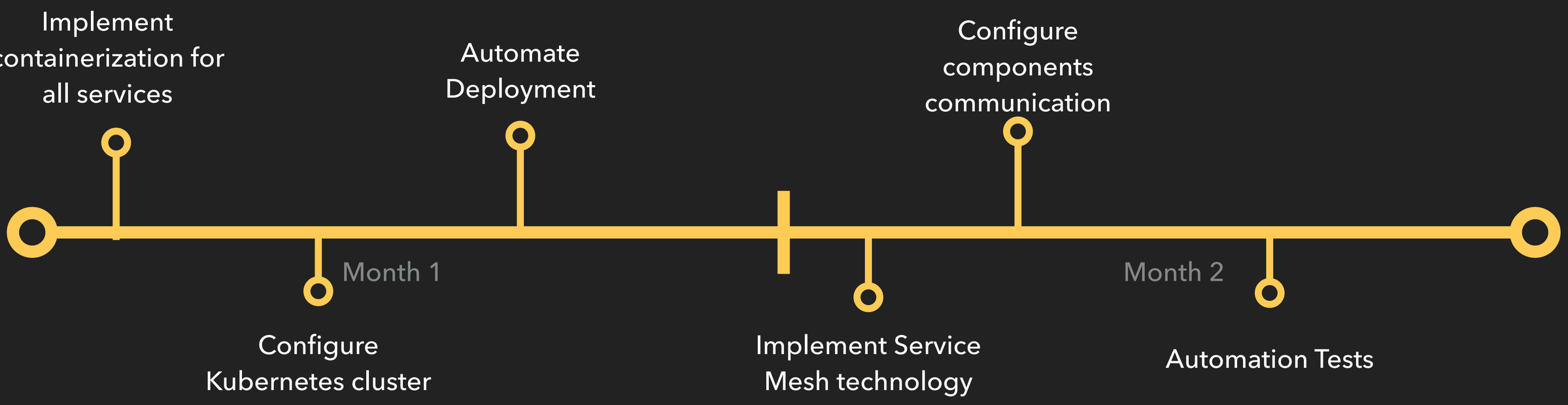
SECURITY

JAPPWARE

Goals

- ✓ Credentials rotation mechanism
- ✓ Key management system
- ✓ Data encryption/decryption

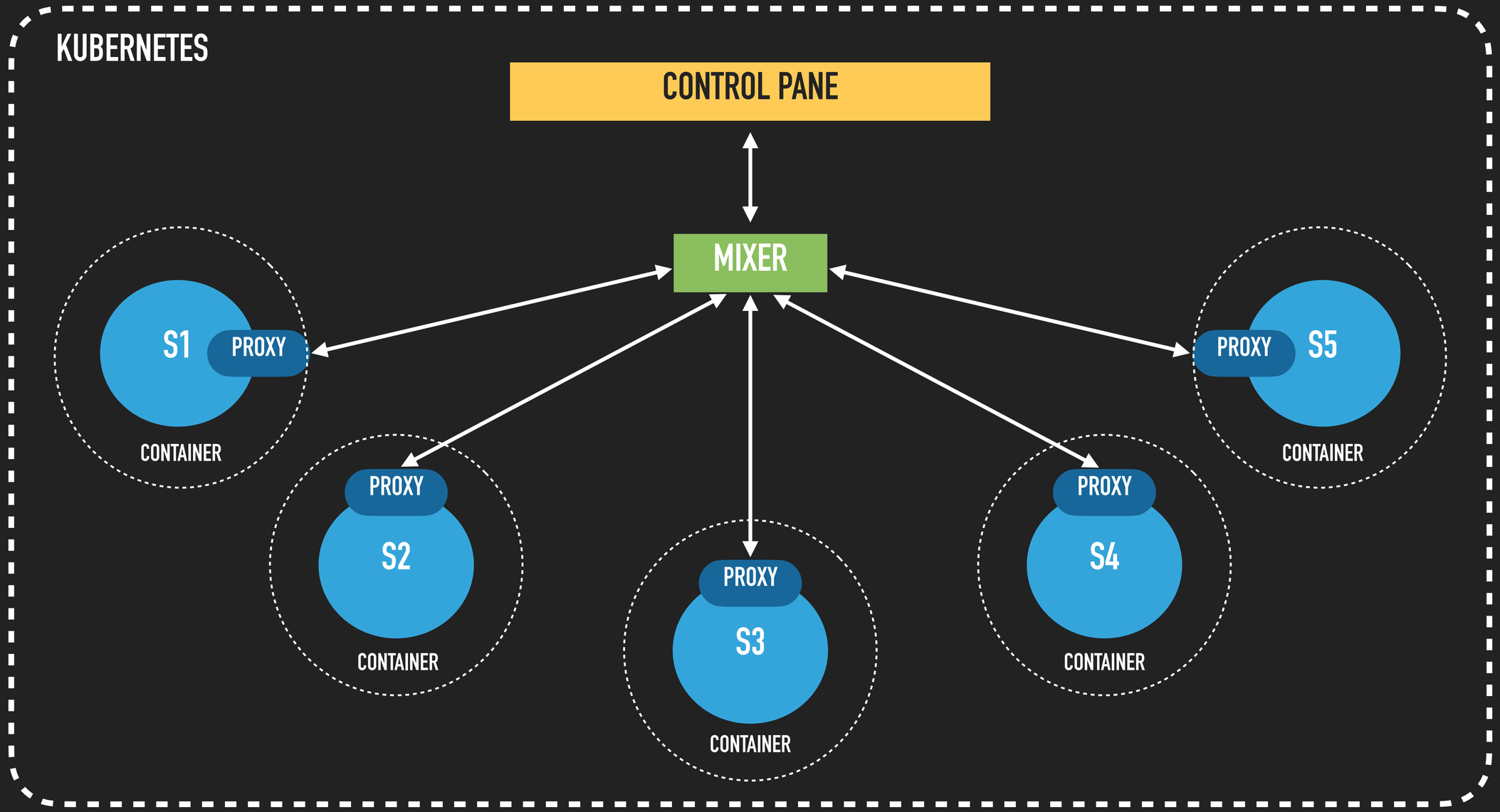
RoadMap



SECURITY

JAPPWARE

Architecture



SECURITY

JAPPWARE

Actions

- ✓ Build Kubernetes cluster and Infrastructure As a Code solution
- ✓ Improve services using Istio proxies for component communication through service mesh channel
- ✓ Configure Service Mesh components to protect internal communication and regularly rotate SSL certificates

Results

- ✓ Secured internal and external communication with the system and between internal components
- ✓ An automated way to run infrastructure on Kubernetes cluster
- ✓ Security check (OWASP top 10) through automation tests

THANK YOU!

JAPPWARE

INNOVATION IN EVERY STEP

info@jappware.com

+38 (063) 189 63 08