GOOD API DESIGN

Zdenek “Z” Nemec
goodapi.co

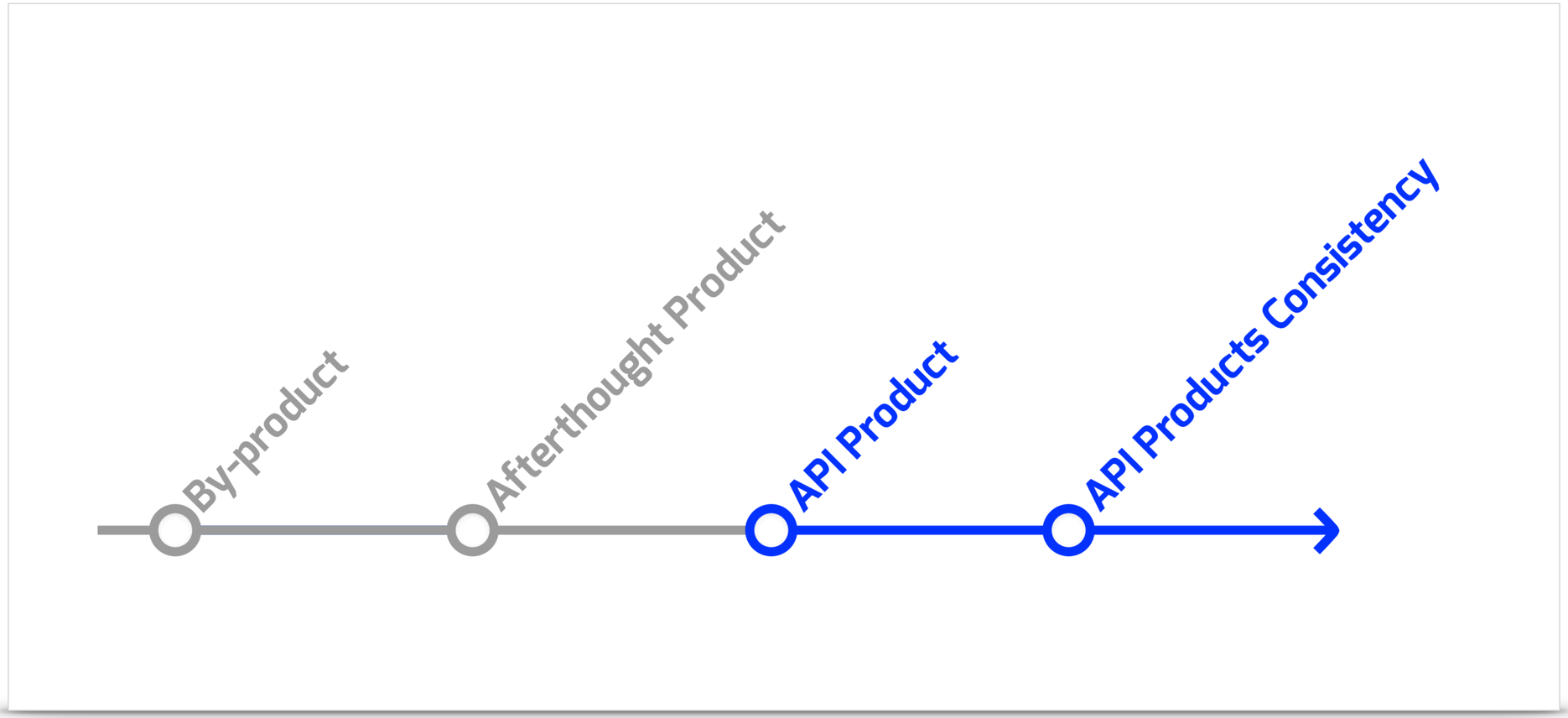
I help businesses build APIs

Zdenek “Z” Nemec

zdne1 @zdne

HOW TO DESIGN AN API?

API DESIGN MATURITY



API IS A PRODUCT

API AS A PRODUCT

- If API is a product treat it as a product
- API-first
- Design-first
  - Enabled through **API description**

API DESCRIPTION

- Human & machine readable description of the API
- Should be accessible to everyone
  - Stakeholders, developers, consumers, tech writers, DevOPS, support …
- Versioned in VCS
- API description represents product requirements
- Approved API descriptions becomes **the contract**

CONTRACT-DRIVEN API LIFECYCLE

THE CONTRACT IS FOLLOWED BY EVERYONE IN THE API LIFECYCLE

API LIFECYCLE PHASES

1 DESIGN

- Collect business requirements
- Identify Affordances & Resources
- Reconcile Vocabularies
- Pick architectural style
- Author API description
- Verify API description
- Check design consistency
- Approve API description - **contract**

2 DEVELOP & TEST

- Implement the API surface
- Test the API locally
- Verify the **contract** locally

3 DEPLOY

- Setup CI/CD pipeline
- Test the API in CI/CD
- Verify the **contract** in CI/CD
- Deploy the API
  - On-prem (Internal Cloud)
  - Public Cloud

4 PUBLISH

- Expose the API internally or externally based on the **contract**
- Security, throttling, consumption setup
- Operational & Business analytics setup

5 OPERATE

- Operational analytics
- Monitor security, throttling, consumption, availability, performance
- Scale
- Support users
- Resolve operational issues

6 CONSUME

- Publish API on Dev portal
- API documentation based on the **contract**
- Obtain API credentials (keys, tokens)
- Experiment with API
- Use the API
- User support

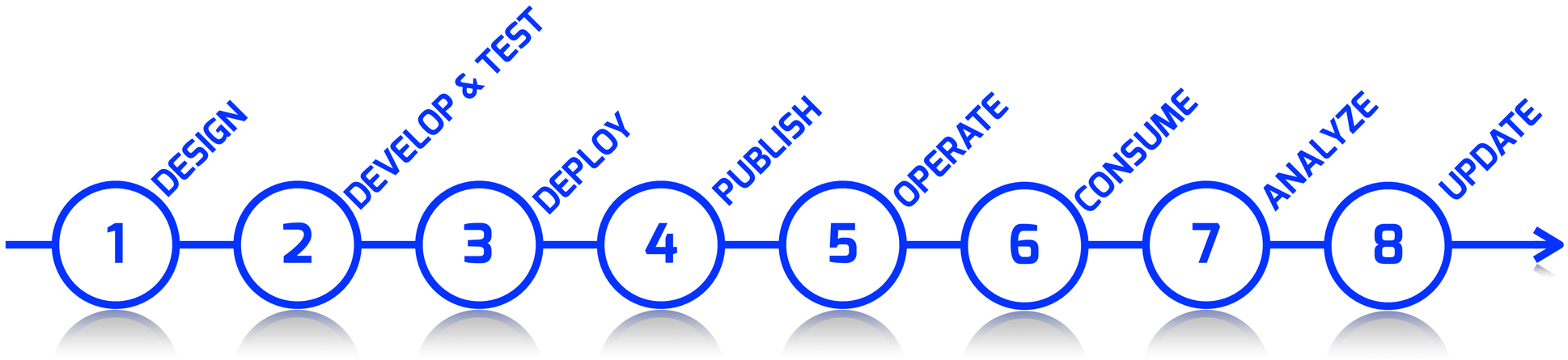
7 ANALYZE

- Business analytics
- Monitor per account usage, **contract** compliance
- Monetization

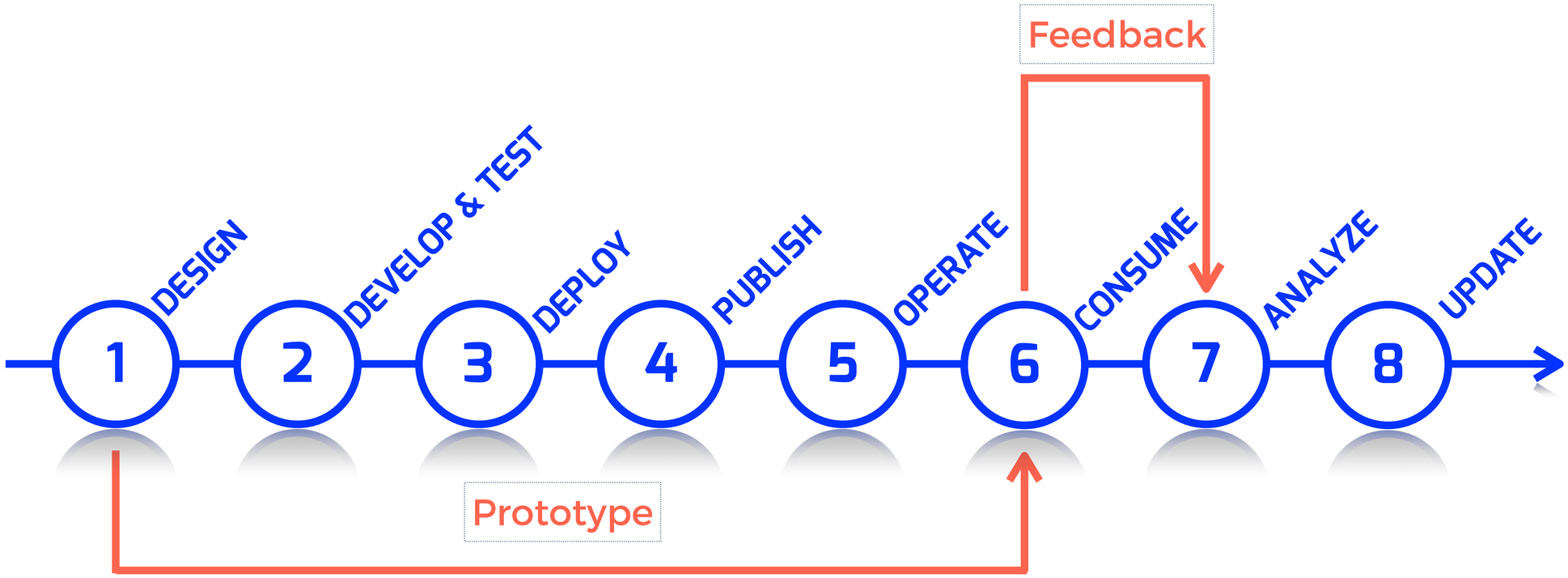
8 UPDATE

- Review the design
- Collect new business requirements
- Update the API description
- Approve the updated API description - **new contract**

WATERFALL


1
DESIGN
2
DEVELOP & TEST
3
DEPLOY
4
PUBLISH
5
OPERATE
6
CONSUME
7
ANALYZE
8
UPDATE

AGILE LIFECYCLE



1

API DESIGN PHASE

1

DESIGN PHASE STEPS

1. **Collect business requirements, use cases**

   green field vs. brownfield scenario

2. **Understand, Reconcile and Define Vocabularies**

   domain-driven design

3. **Pick architectural style, decide on the protocol**

   role of the API architect

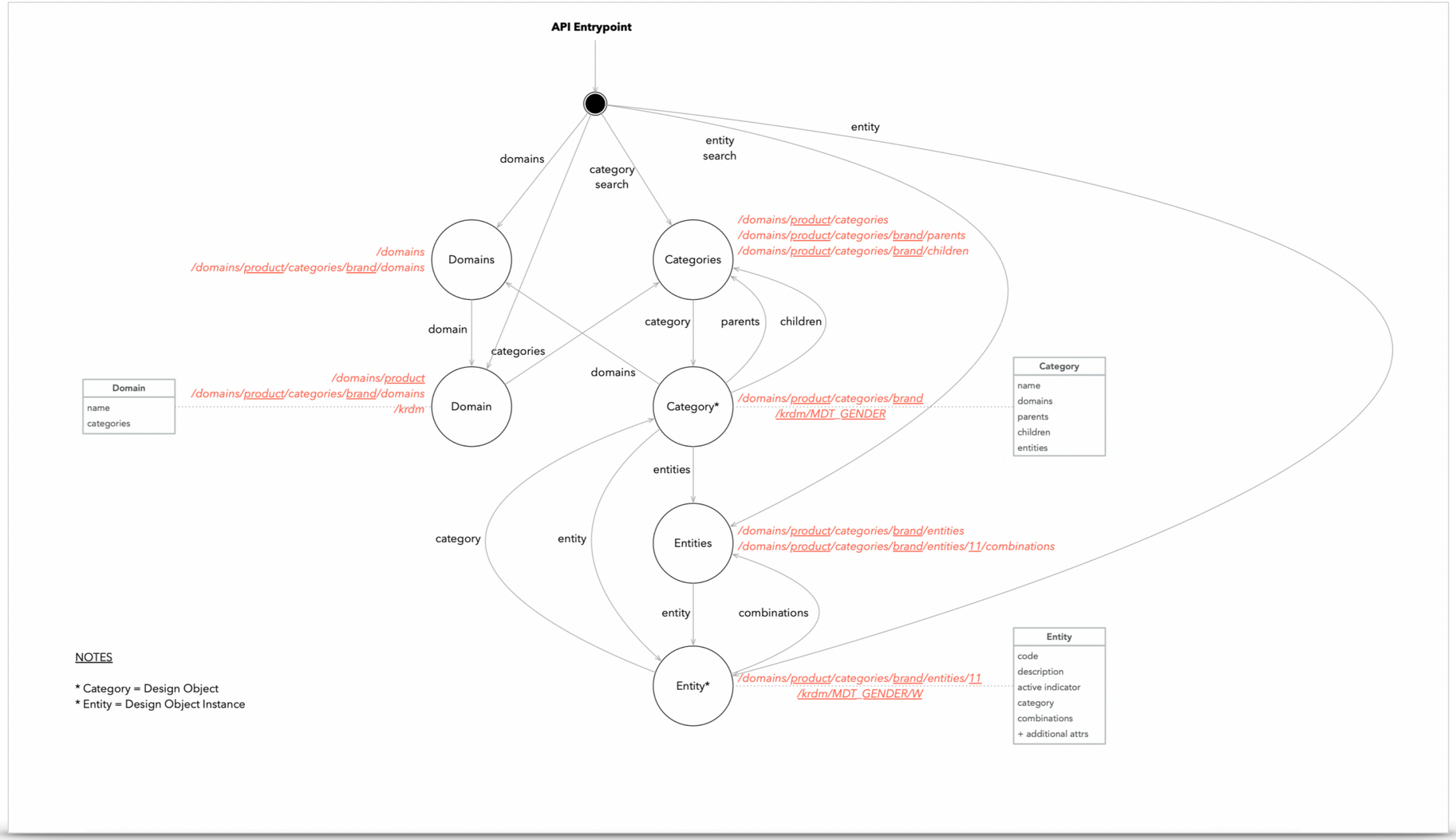
1

WEB APIS

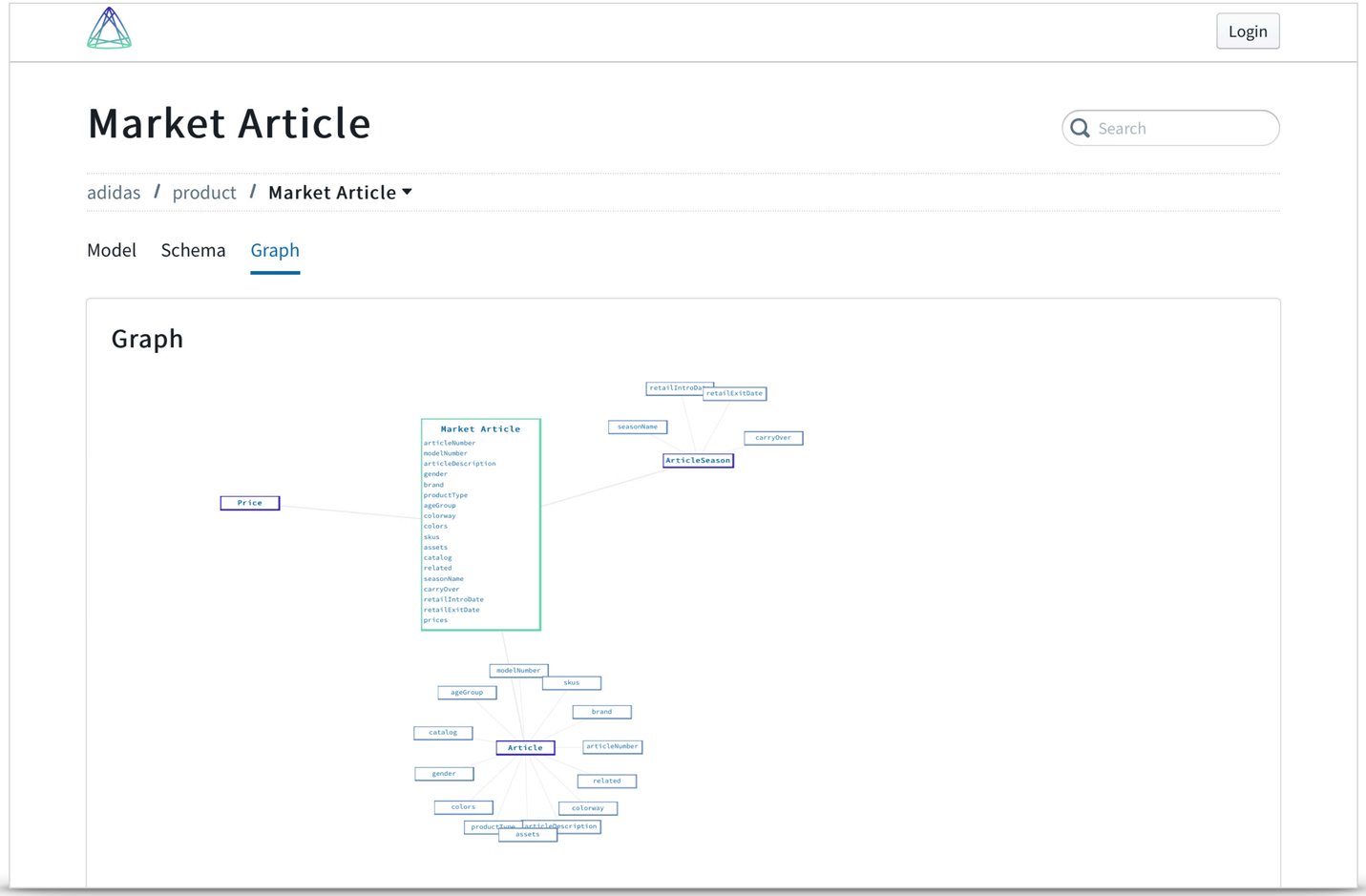
4.Web APIs

1. Identify Affordances & Resources
2. Draw Finite State Machine Diagram
3. Author API Description
    1. Define resources & actions
    2. Define models in Supermodel

API AUTOMATON



SUPERMODEL DATA MODEL



https://supermodel.io

1

GRAPHQL APIS

5.GraphQL APIs

1. Identify core queries
2. Identify mutations
3. Define models in Supermodel
4. Author GraphQL Schema

1

DESIGN PHASE STEPS

6.Verify API Description

developers, clients & stakeholders

7.Check API Design consistency

style-guides & design patterns, design governance

8.Approve the API Description–contract

binding for every party in API lifecycle

9.Evolve the design – Update Phase

API PARADIGMS
API ARCHITECTURAL STYLES

API PARADIGMS

Web APIs
REST

Query APIs
GraphQL

Streaming APIs
Kafka

RPC APIs
gRPC

Web APIs
HTTP

Flat File

RPC APIs
SOAP

THE DECISION SHOULD ALWAYS BE FUNCTION OF YOUR

CONSTRAINTS

CONSTRAINTS

1. **Distributed System Properties**

   execution qualities of the system we aim to build

2. **Business Constraints**

   customer-related and business requirements

3. **Complexity Constraints**

   implications of distributed system complexity, evolution qualities

4. **Domain Constraints**

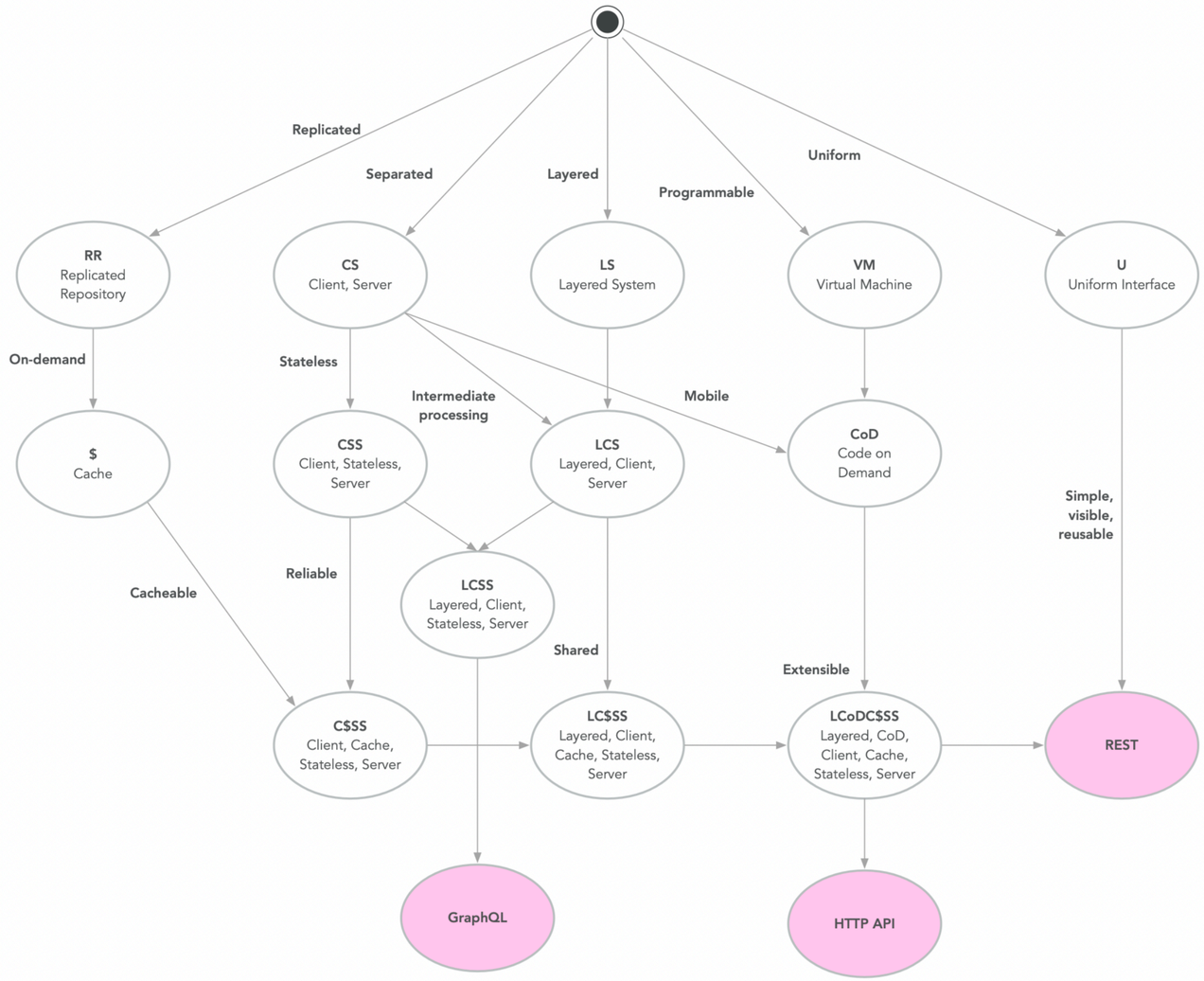
   domain-specific limitations, regulations

5. **Cultural Constraints**

   Conway's law, knowledge, resources, hype, trendiness

> *"organizations which design systems ... are constrained to produce designs which are copies of the communication structures of these organizations."*
>
> *— M. Conway*

DISTRIBUTED SYSTEM PROPERTIES



Three Waves of APIs

First Wave (2000)

Customer-Specific APIs

one-to-one: few large established customers

Second Wave (2010)

Generic APIs

one-to-many: many mid- or small- size customers

Third Wave (2020)

Autonomous APIs

many-to-one & machine-to-machine: automatic, later autonomous APIs


COMPLEXITY

COMPLEXITY OF DISTRIBUTED COMPUTING SYSTEMS

Task-structure Complexity

How difficult it is understand how to perform a task in a distributed system

Unpredictability

How difficult is to predict effect of an action in distributed system, amount of entropy

Size Complexity

Size of the system implies cognitive complexity, large number of concepts and the knowledge required

Chaotic Complexity

Small variations in a certain part of the system can have large effects on overall system behavior

Algorithmic Complexity

Both traditional algorithm complexity in the term of time and space and cognitive complexity of understanding algorithm

IF THE NUMBER OF

API Designs x Versions x Deployment Hosts x Clients x Client Deployments = 1,10565×10^9

130 APIs

70 versions

3 x 6 (environments x availability zones)

1500 integrations

own deployments: 3 x 6

on-premise deployments: 1000

apps: 3 000 000

IoT: ?????

IS HIGHER THAN TWO

You are likely to face complexity using, maintaining and evolving your API

Solutions to Complexity

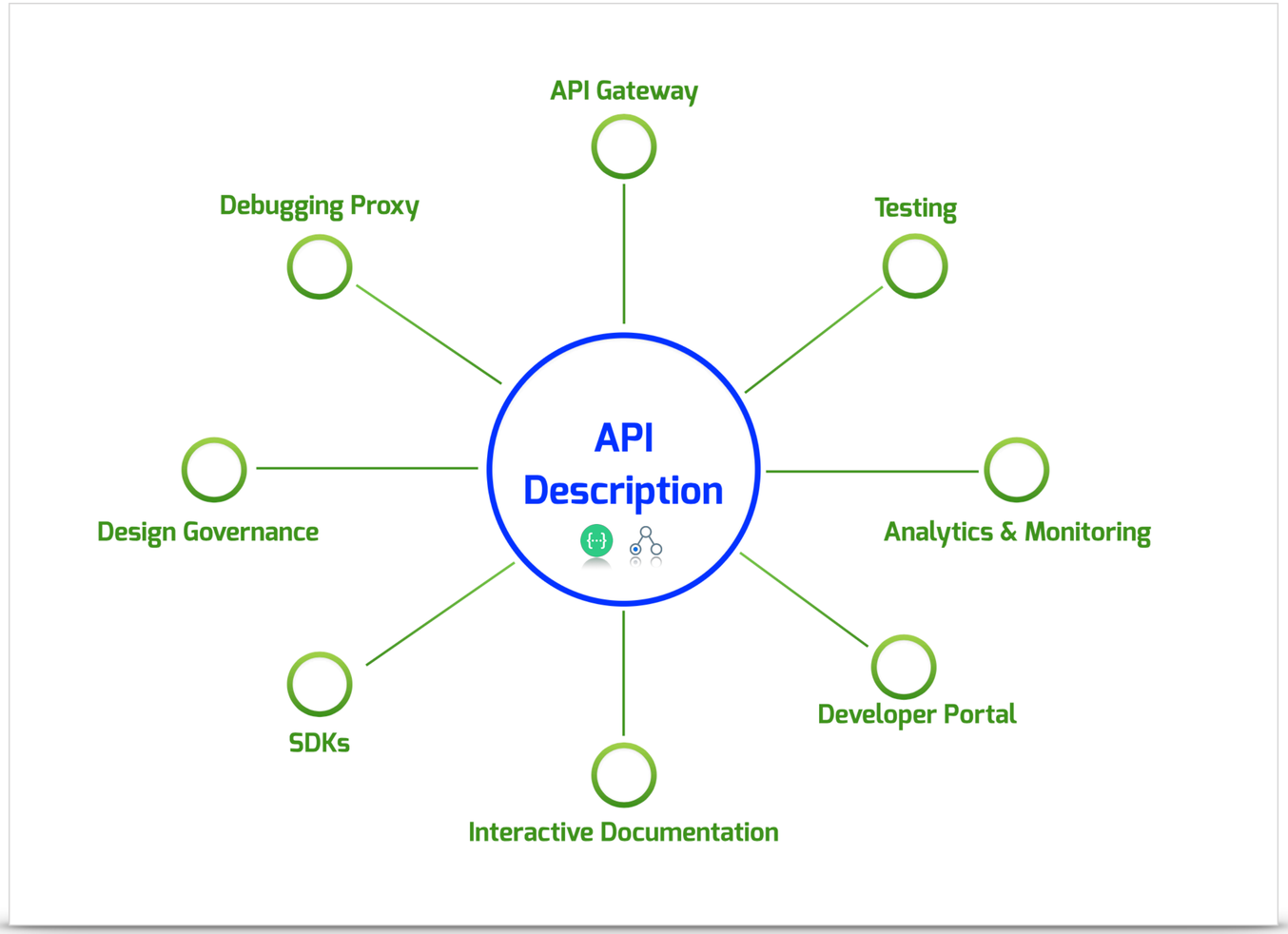
1. Hire more people
2. Standardization
3. Apply massive automation
4. Start building distributed systems differently–change the approach–autonomy of the components in API landscape

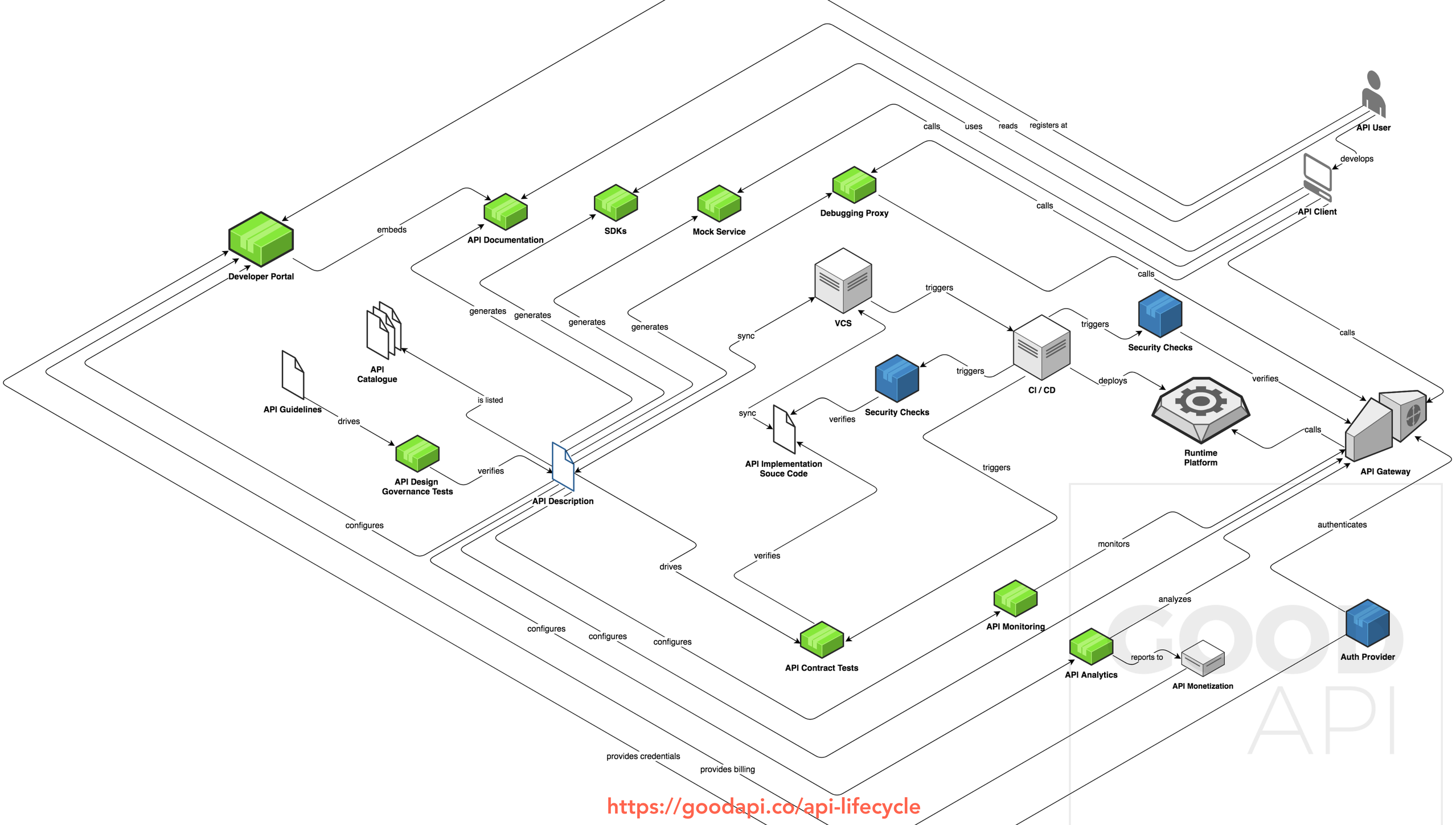
AUTOMATED
CONTRACT-DRIVEN API LIFECYCLE

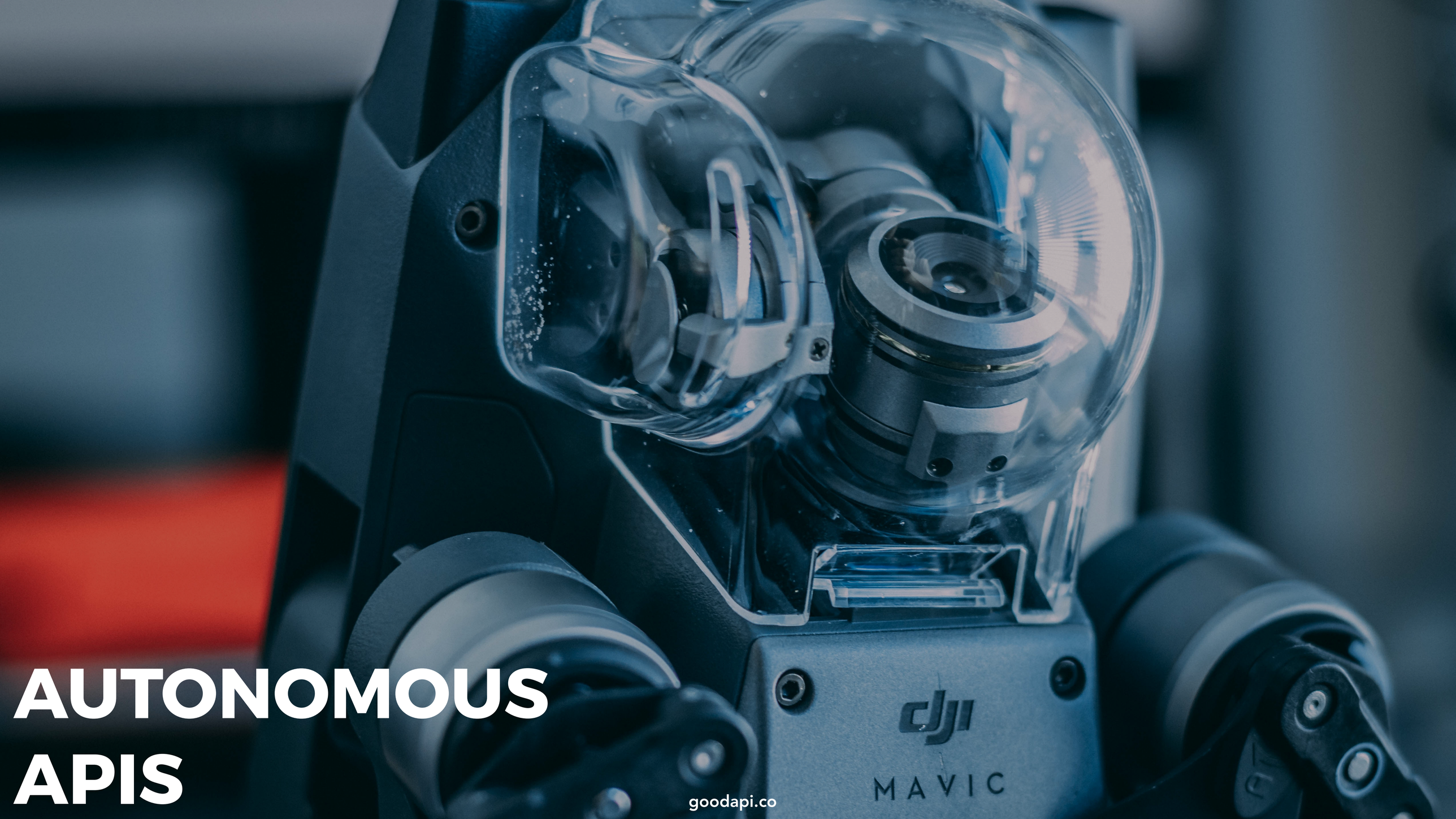
747 Flight Engineers

Early four-engine jets required flight engineers. Later four-engine jets were designed with sufficient automation to eliminate the position.

CONTRACT-DRIVEN




API User
registers at
reads
uses
calls
develops
API Client
Developer Portal
embeds
API Documentation
SDKs
Mock Service
Debugging Proxy
calls
generates
generates
generates
generates
sync
VCS
triggers
calls
API Catalogue
is listed
API Guidelines
drives
API Design Governance Tests
verifies
API Description
configures
sync
API Implementation Souce Code
verifies
Security Checks
triggers
CI / CD
triggers
Security Checks
deploys
Runtime Platform
verifies
calls
API Gateway
triggers
drives
verifies
configures
configures
configures
API Contract Tests
monitors
API Monitoring
analyzes
API Analytics
reports to
API Monetization
authenticates
Auth Provider
provides credentials
provides billing
GOOD API
https://goodapi.co/api-lifecycle



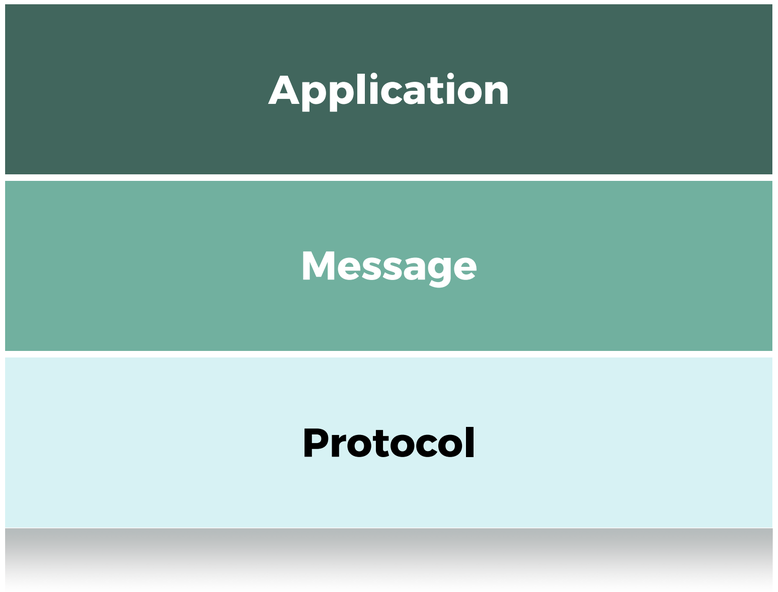
AUTONOMOUS APIS


GOOD
API
INDEPENDENT API CONSULTING

IMAGES BY UNSPLASH.COM

BACKUP

API SEMANTIC LAYERS



Application Domain Semantics
(e.g. Hospitality domain)

HAL Semantics

HTTP Protocol Semantics