Automation API software

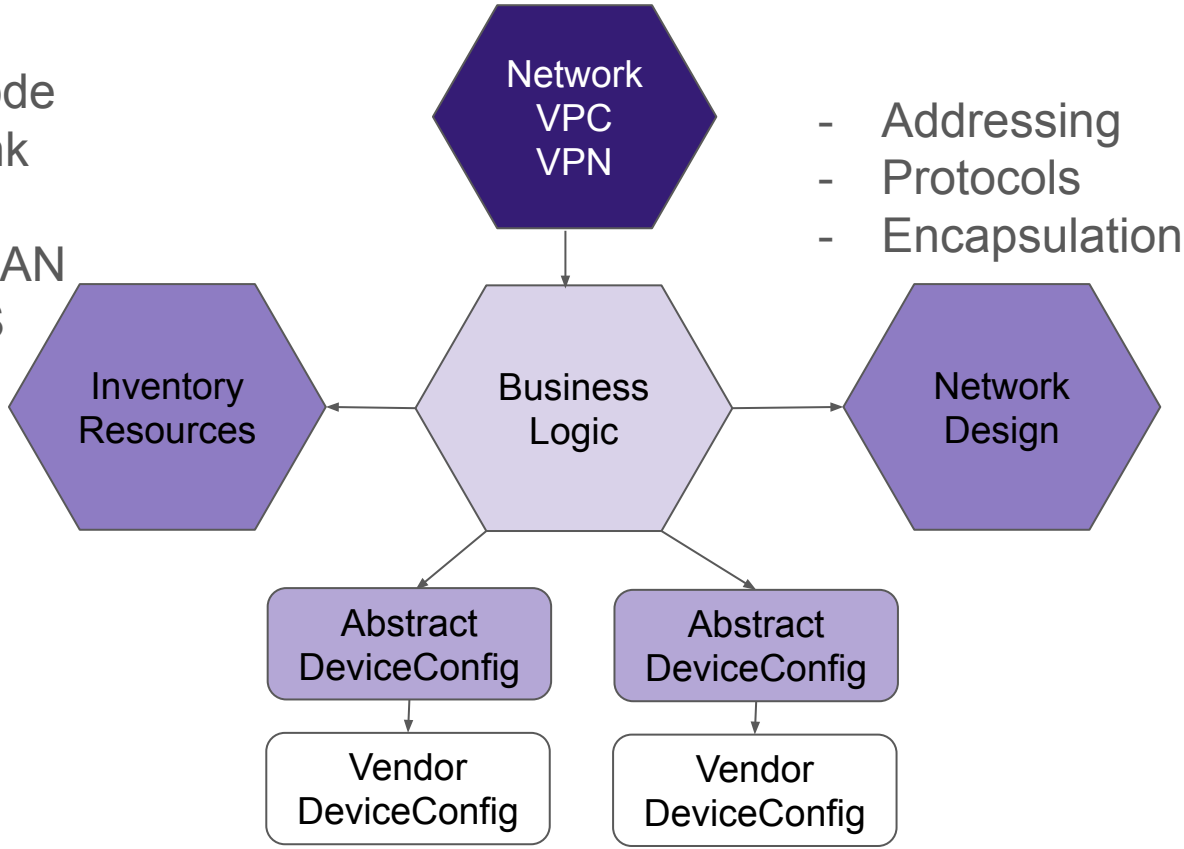
Wim Henderickx - Nokia

Introduction

- The rise of declarative automation tools/systems
  - Terraform: IaaC tool
  - Kubernetes: container orchestration
- How can we use schema based systems for automation and orchestration?
  - YANG
  - OpenAPI
- Note: While i refer a lot to kubernetes principles in this presentation - most of the content is not dependent on kubernetes (although you can deploy this within a kubernetes cluster)

The big picture

- Node
- Link
- IP
- VLAN
- AS



- Addressing
- Protocols
- Encapsulation

Abstraction

Business Logic

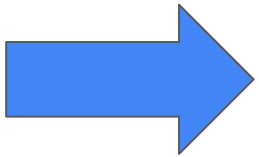
Normalization

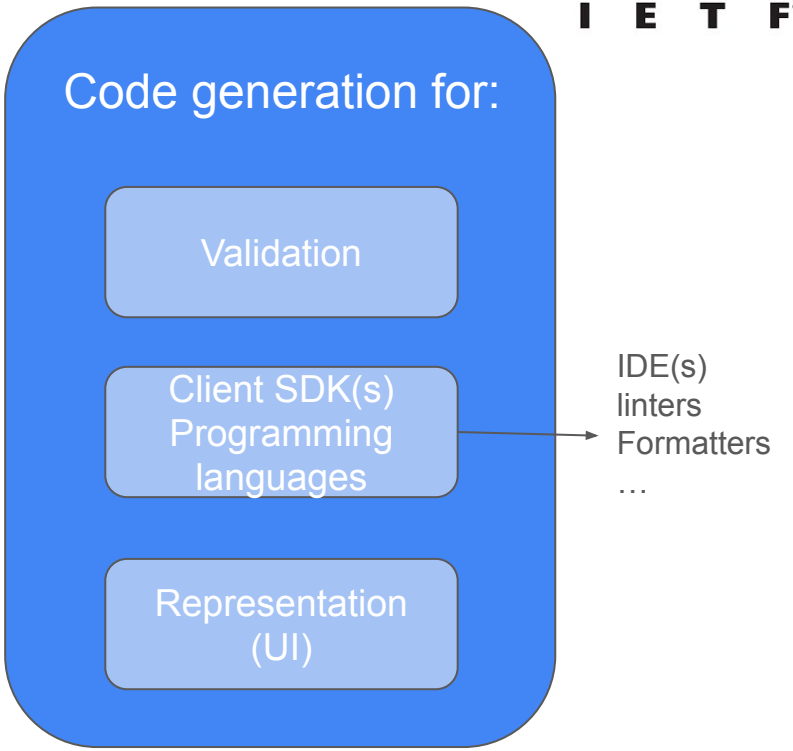
Business Logic

Vendor specific
Implementation

The goal of the Schema

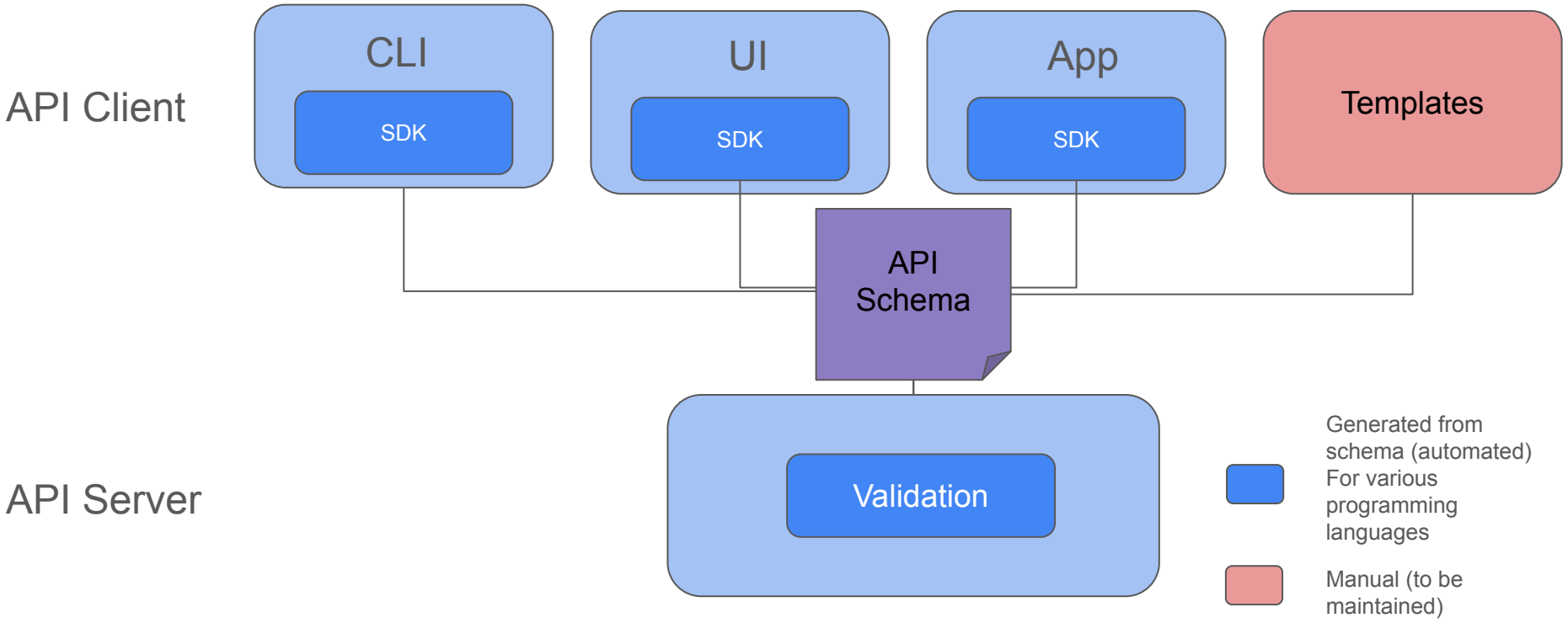
- Schema
  - Capabilities
  - Parameters and types
  - Constraints
  - Relationships
- API operations
  - Imperative/Declarative
  - REST
  - GraphQL
  - GRPC
  - Netconf/gNMI
  - Transactions
  - Automation support (finalizers, dependents, …)
  - Pub/Sub (OnChange/Event driven)





Offload developers from the plumbing -> focus on the application business logic

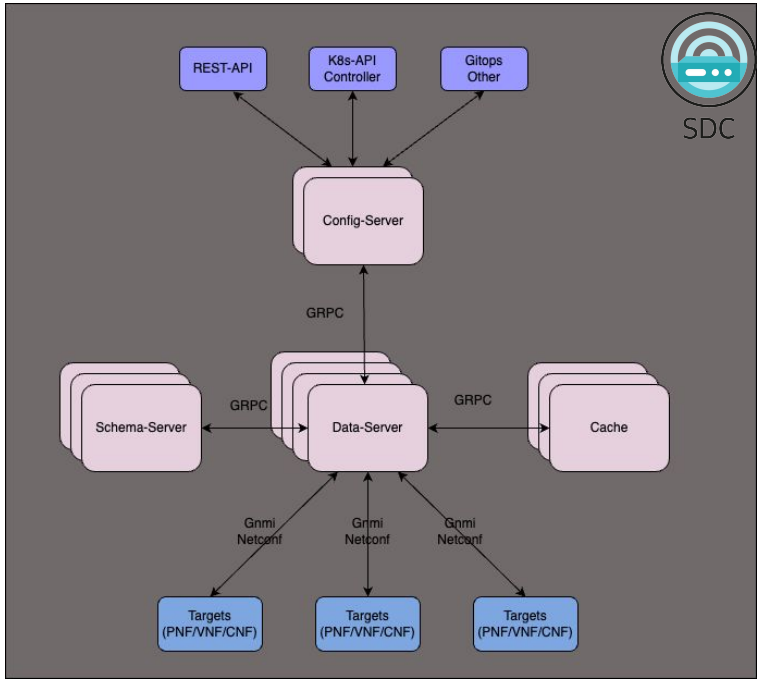
What does a schema enable?



Introducing SDC (Schema Driven Configuration)

https://docs.sdcio.dev

- Open source
- Declarative/idempotent
- Focussed on YANG schema
- gNMI and Netconf
- Config and Config snippets
- Conflict management (Layered)
- Subscription
- Validation/Drift/Deviation
- DryRun, Config Blame, Orphan, …
- Multi-vendor
- PNF, VNF, CNF, NooP
- Config and State



SDC enables YANG based system to be consumed from kubernetes API

SDC Example

Config/ConfigSet

SDC

```
apiVersion: config.example.com/v1alpha1
kind: Config
metadata:
 name: test
 namespace: default
 labels:
   config.sdcio.dev/target Name: dev1
   config.sdcio.dev/targ etNamespace: default
spec:
 priority: 10
 config:
 - path: /
   value:
     interface:
     - name: "system0"
       admin-state: "enable"
       description: "k8s-system0"
```

META

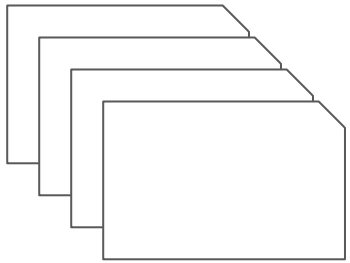
USER

YANG

Device

Device

Config/ConfigSet snippets

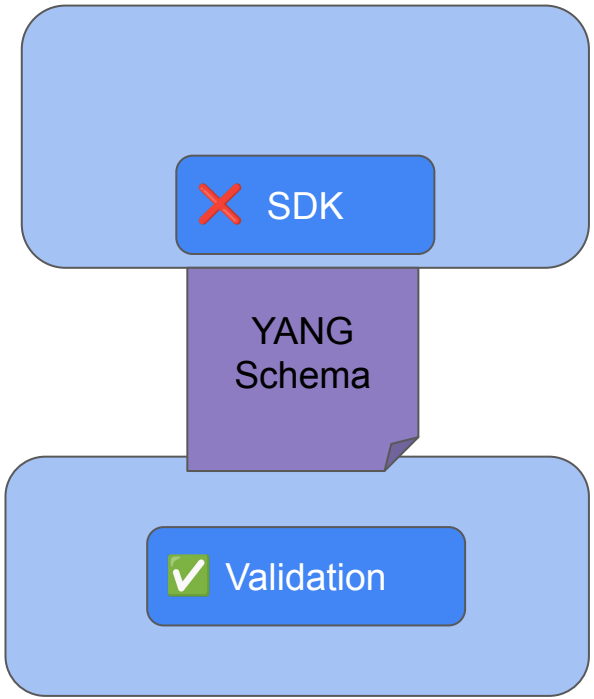


SDC

Device

Device

SDC state of affairs

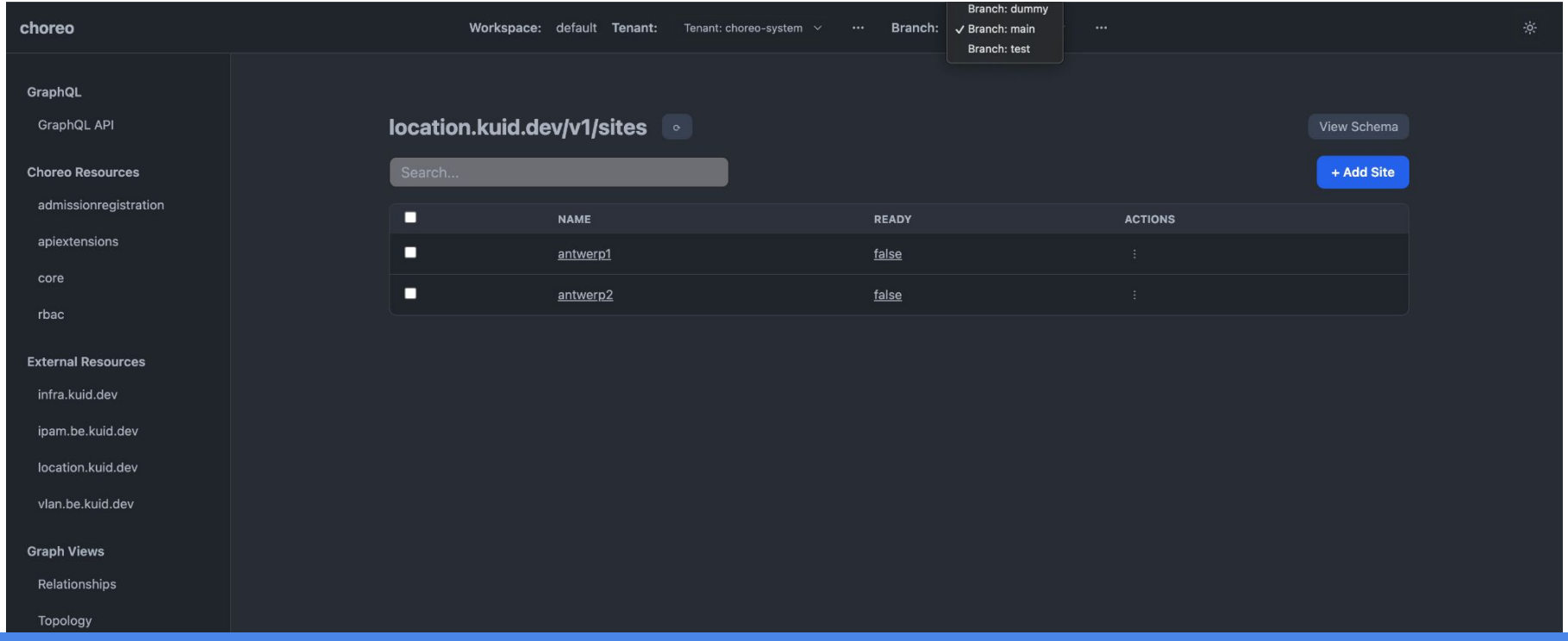


- Work in progress
  - SDCsim (summer 2025)
  - Network wide transaction (summer 2025)
  - SDK generation (bachelor TBD)
    - Python
    - Go
    - TypeScript
    - Rust
    - …
  - CLI (bachelor TBD)
  - YANG push (TBD)
  - …
- YANG:
  - Very few open source tooling
  - Bugs in vendor models when validating with data.

API Operations that support/simplify automation

- API Operations that simplifies your automation
  - CRUD operation (REST/GraphQL)
  - PATCH
  - Atomic/Transaction
  - Conflict Management: allows for concurrent access to the API by multiple actors
  - Selectors/Filters: flexible query parameters
  - Bulk Operations: e.g. Delete
  - Declarative/Idempotent but allowing imperative operations.
  - Event/integrated Pub/Sub (OnChange) with selectors/filters
  - Finalizers: don’t delete unless the dependent finishes the job
  - Owner References: Who owns the data - allow config blame
  - Relationships/Graph: how is your data connected - relationships/dependencies
  - Dry Run/Version Control
  - …

Introducing c4o (choreo)



Goal: Have developers focus on: schema/data model + business logic

c4o (choreo) Example

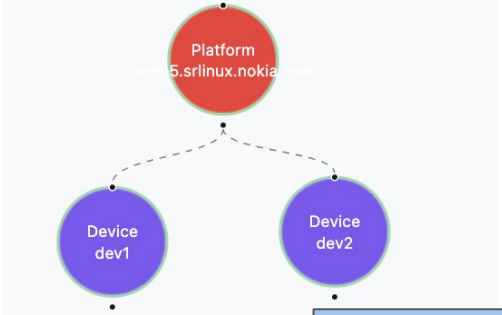
Schema

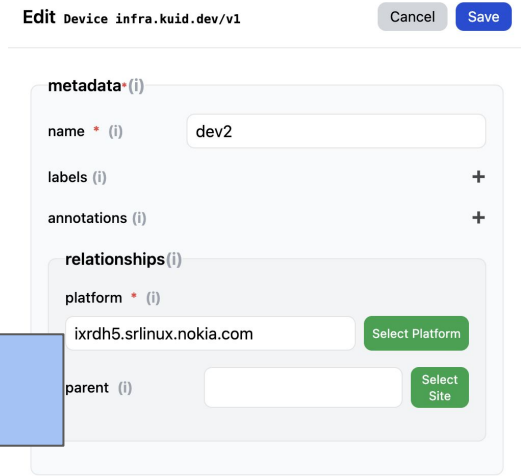
```yaml
- name: infra.kuid.dev
  apis:
  - resource: devices
    kind: Device
    scope: workspace
    versions:
    - name: v1
      storage: true
      attributes:
      - name: spec
        kind: object
        attributes:
        - name: name
          kind: text
          required: true
          description: "name of the device"
      relationships:
      - name: platform
        target:
          group: infra.kuid.dev
          kind: Platform
        cardinality: one
        required: true
      - name: parent
        target:
          group: location.kuid.dev
          kind: Site
        cardinality: one
```

From schema to … (generated)

Relationships

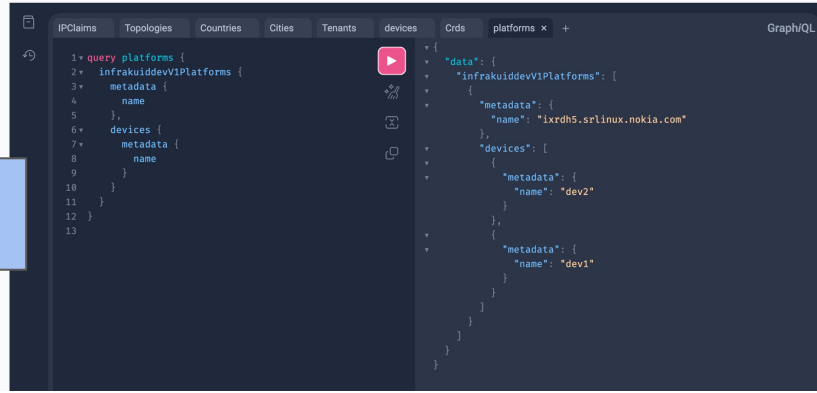
UI



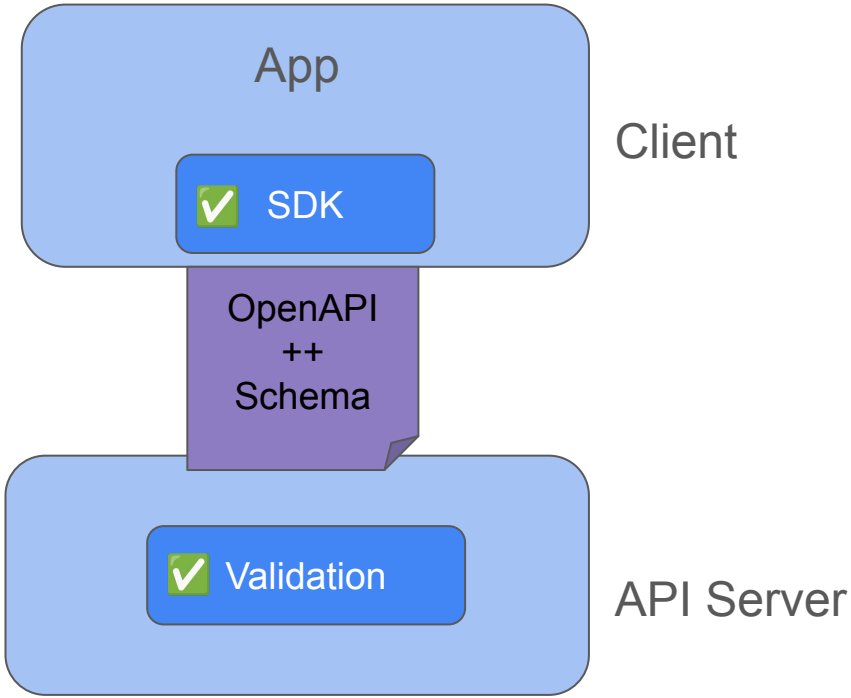


SDK

GraphQL



c4o (choreo) Overview



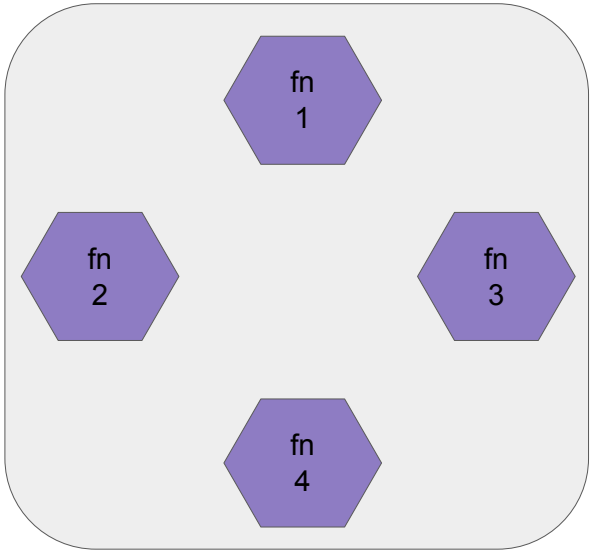
Client

API Server

- An automation engineer
  - Define schema (uses an existing schema)
  - Build your business logic
- Extendable/Pluggable
  - Admission controllers
  - Controller runtime
  - Schema
  - Business logic
- Supports Automation hooks: see slide 9 regarding API operations supporting
- Specific extensions
  - Relationships/graph
  - Version Control
  - GRAPHQL
  - WASM

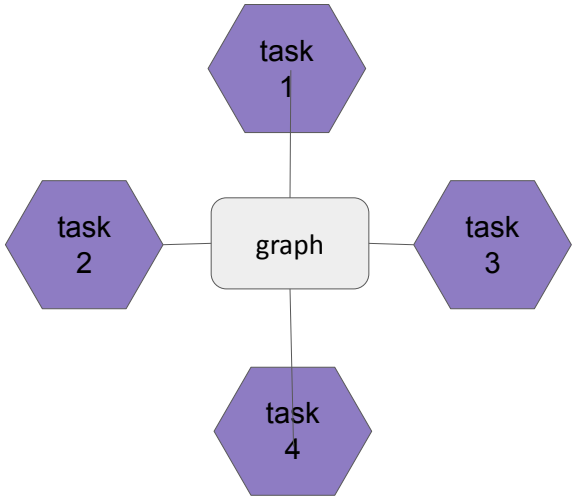
c4o (choreo) automation approaches

Monolith program



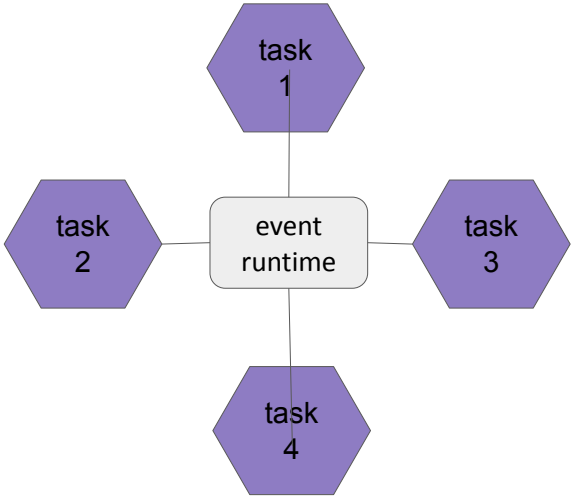
- E.g. a python program

Workflow



- Ansible (predefined workflow)

choreography



- Event driven and distributed

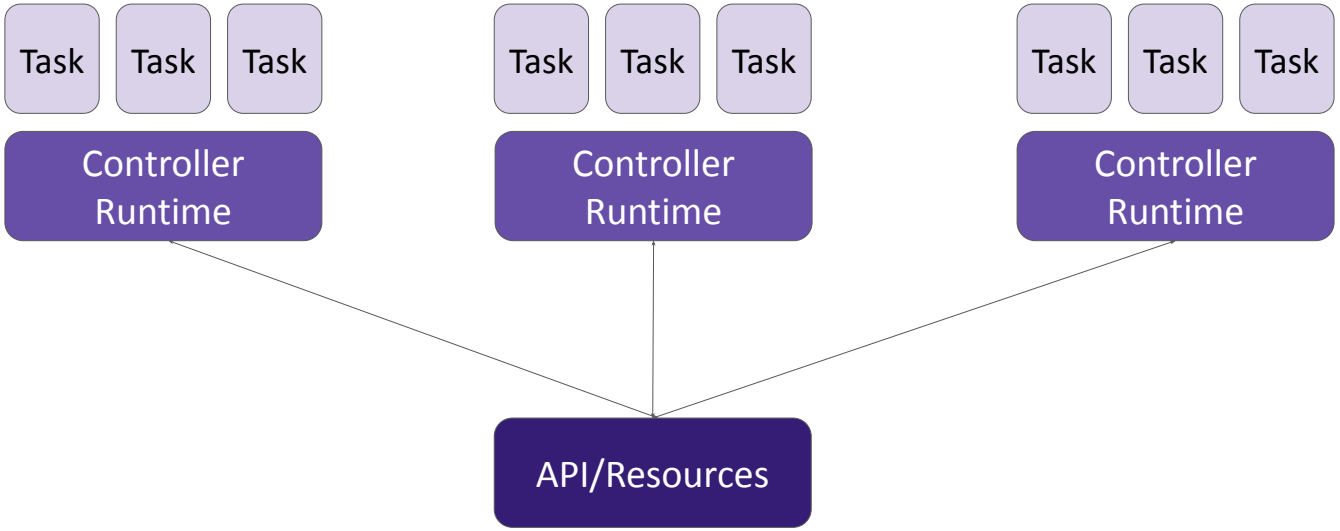
Summary

- Schema
  - Code generation allows developers to focus on the application business logic:
    - Machine generated Validation, Client SDK(s), UI representation
- SDC:
  - Enables declarative operations for YANG based systems (Netconf/gNMI, others TBD)
  - Validates Schema semantics (according to YANG rules)
  - Validates Schema in conjunction with data (according to YANG rules)
  - Drift detection, …
- c4o (choreo):
  - API server/framework that simplifies your automation with version control
  - Flexible schema (User controlled, Code Generators (Validation, SDK, Representation))
  - Business logic -> (ansible, terraform, python, nornir) or using event driven controllers

Feedback welcome/Questions ?

Backup

Distributed microservices based automation



Distributed and decoupled