redhat.

SYNDESIS OPERATORS

Proposal

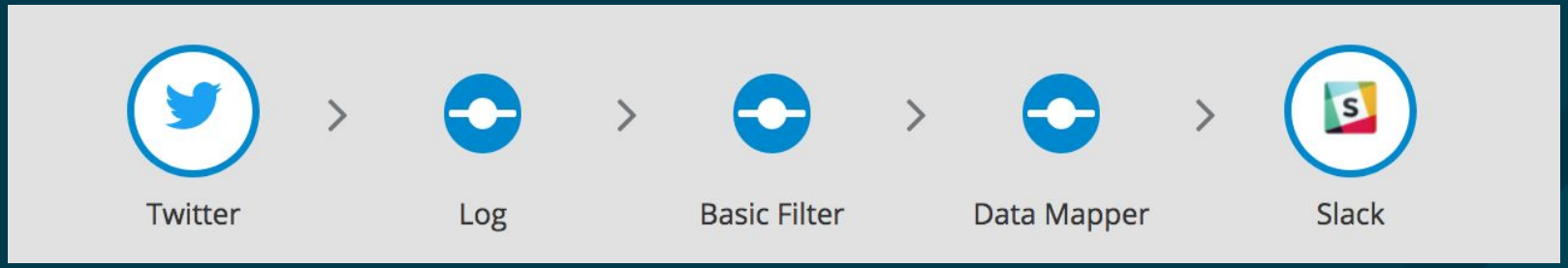
roland

IPAAS

Low-code integration platform as a service

Syndesis	Upstream
Red Hat Fuse Online	Hosted on OpenShift Online & On Premise on OCP
iPaaS	Everything

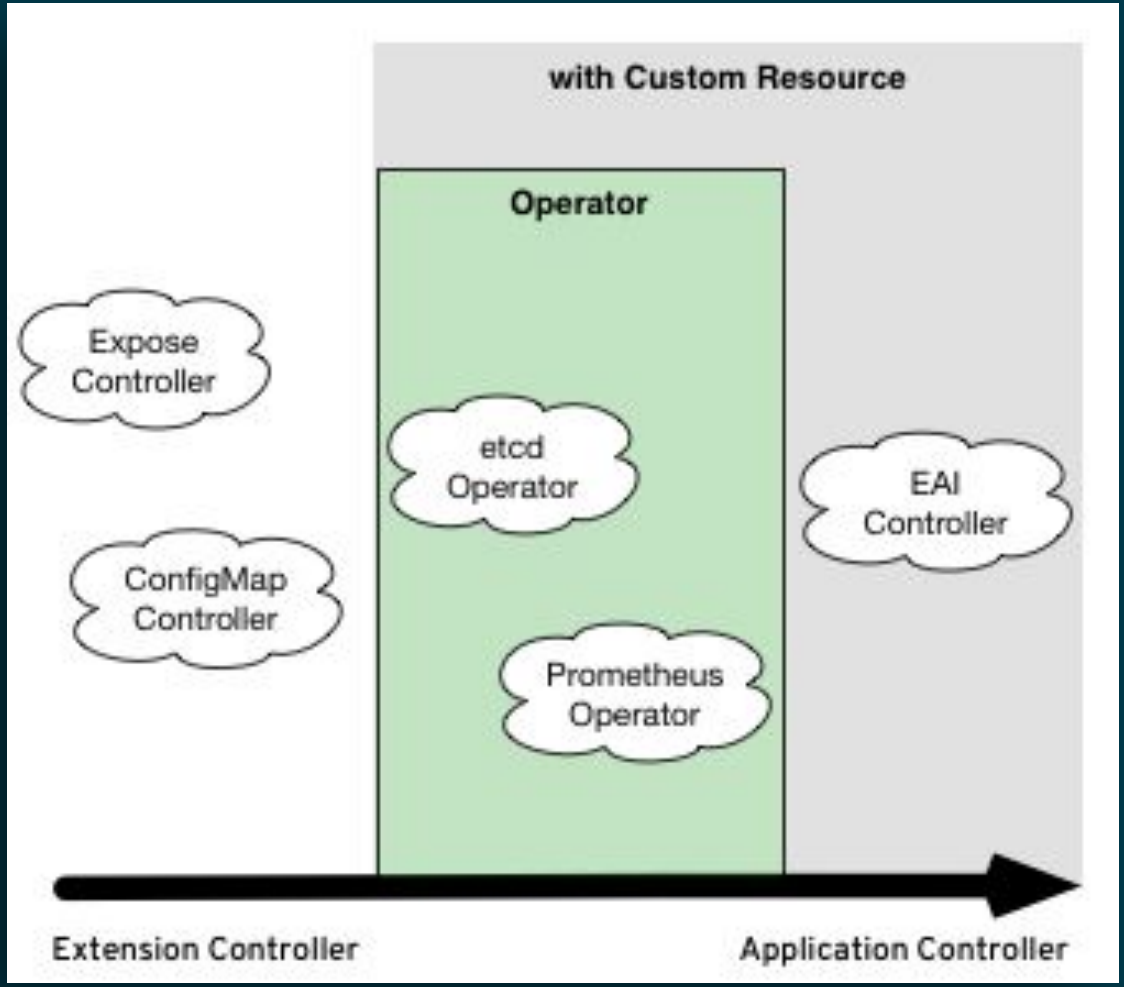
- Part of **Red Hat Fuse 7.0** (released 2018-06-04)
- “Camel in the Cloud”



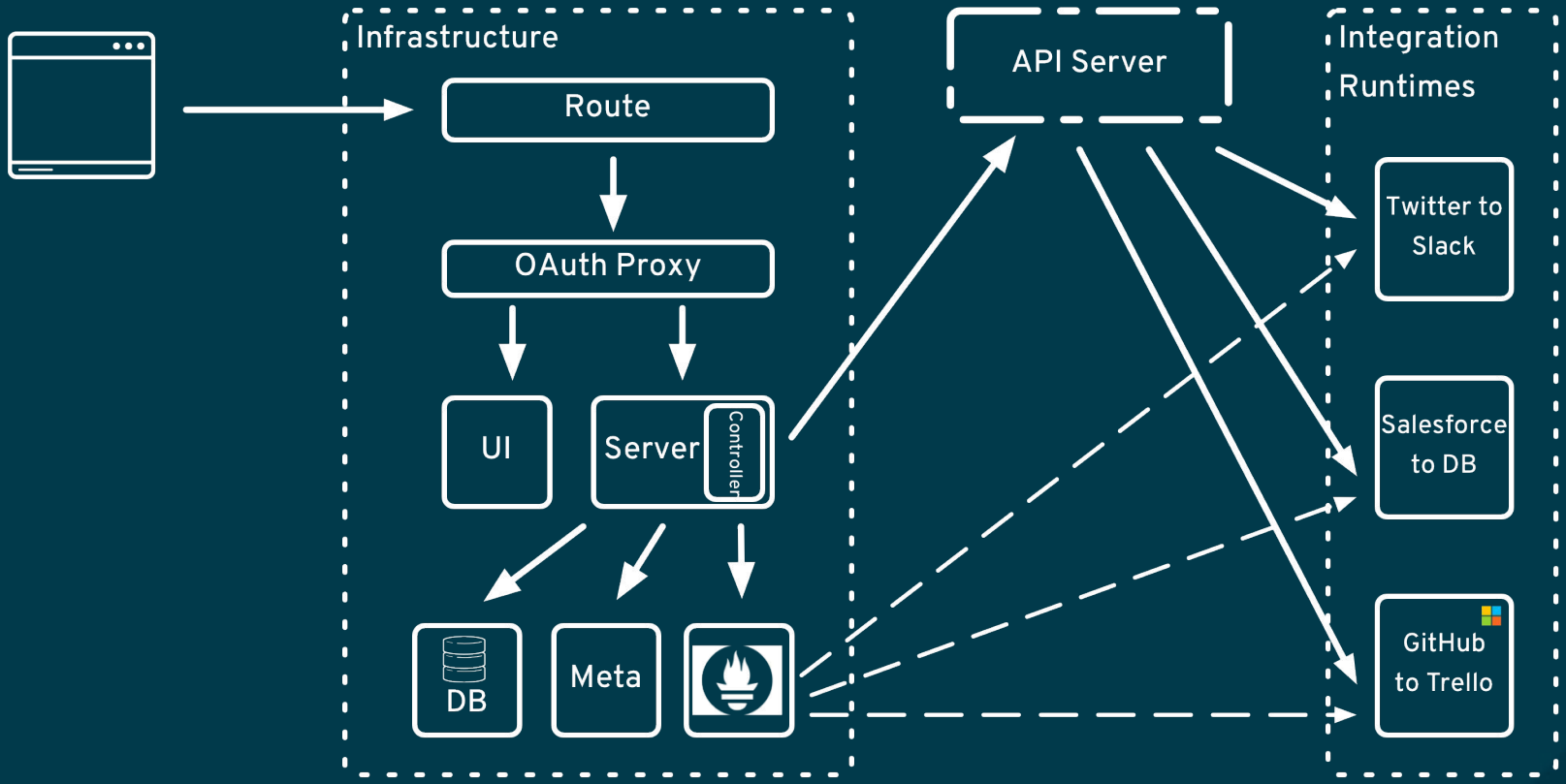
Operators

- “An Operator is a method of packaging, deploying and managing a Kubernetes application”
- **Custom Resource Definition** (CRD) for describing the application
- **Custom Controller** for reacting on such custom resources
- CoreOS’ Operator Framework:
  - **Operator SDK** : Scaffolding and libraries for creating Operators in Golang
  - **Operator Lifecycle Manager** : Installation, updates and management of Operators themselves
  - **Operator Metering** : Usage reporting for operators (to come)

Operators



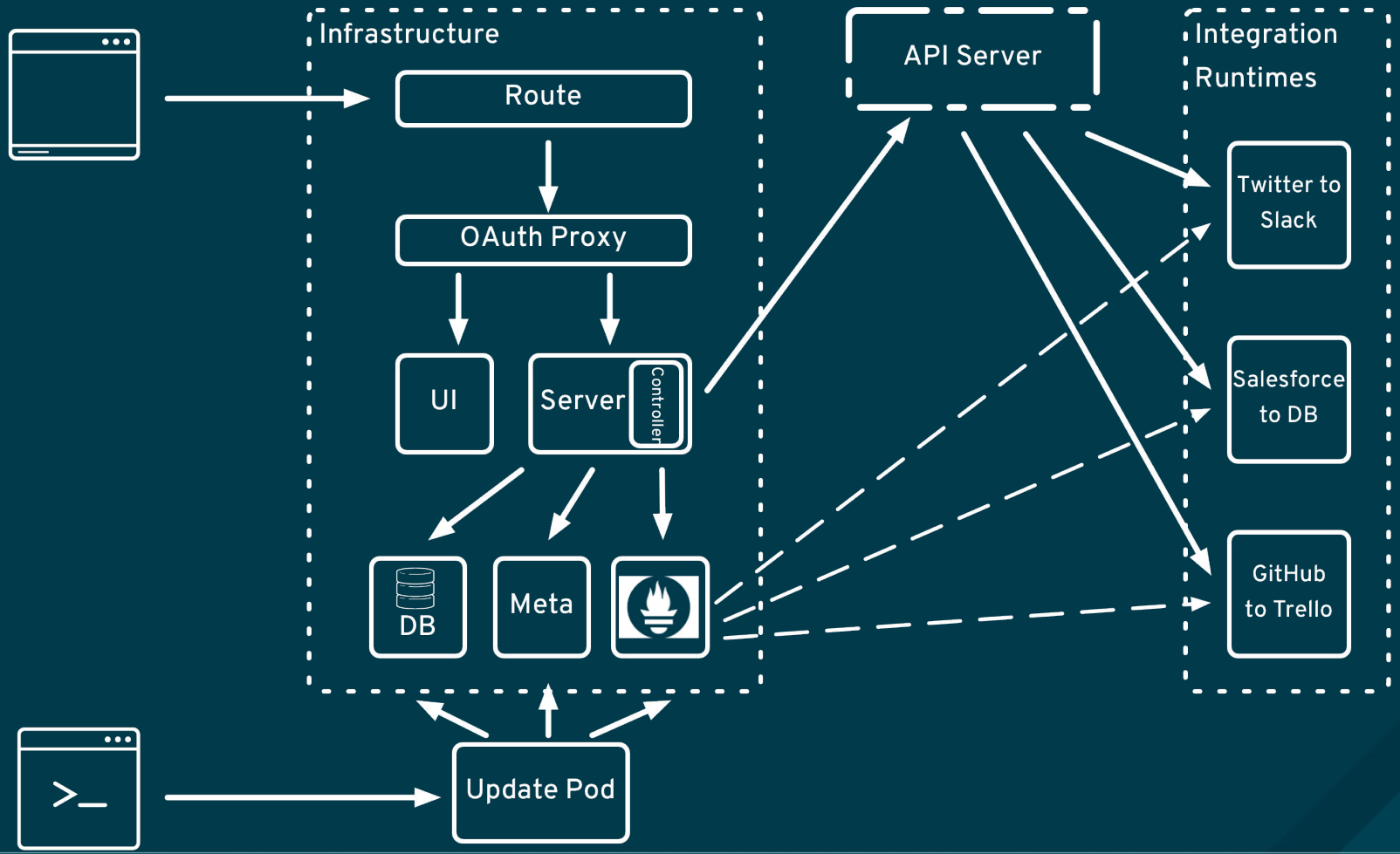
ARCHITECTURE



INFRASTRUCTURE LIFECYCLE

- Create:
  - Installed via OpenShift Template
- Update
  - Dedicated Update Pod which is installed manually on update
  - Set of shell scripts accessing the OpenShift API Server externally via oc
- Delete
  - Deleting all infrastructure objects

INFRASTRUCTURE UPDATE

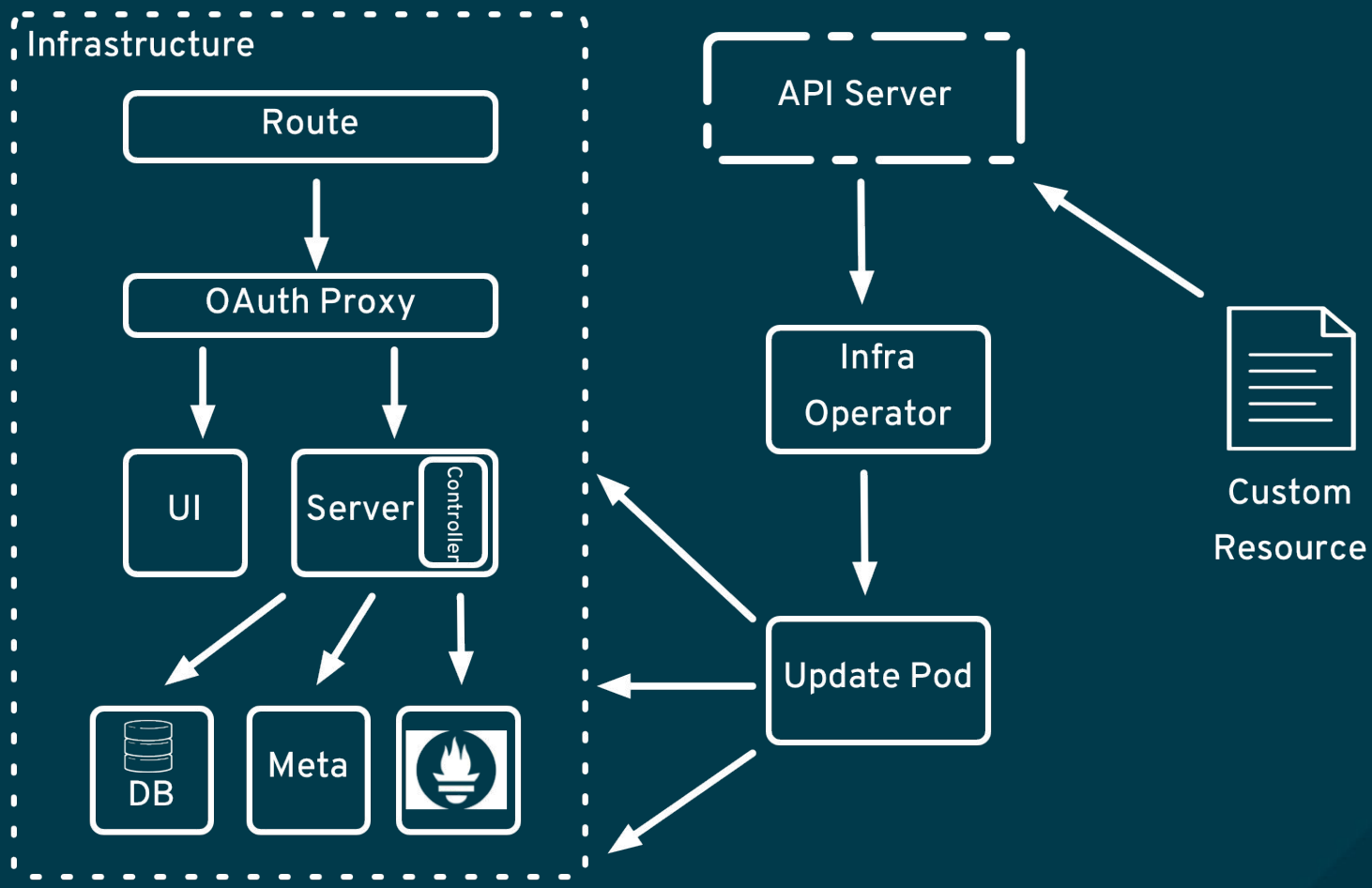


RUNTIME LIFECYCLE

- Create:
  - Within a user HTTP request (sync)
  - S2I Build of Runtime Image created and started
  - Deployment / DeploymentConfig (1 replica) created
- Status check
  - Periodic polling (1 min schedule)
- Delete
  - On user request (sync)
- Update
  - Delete and Re-create
  - No history

INFRASTRUCTURE OPERATOR

INFRASTRUCTURE OPERATOR



INFRASTRUCTURE OPERATOR

Replace Installation and Update Process

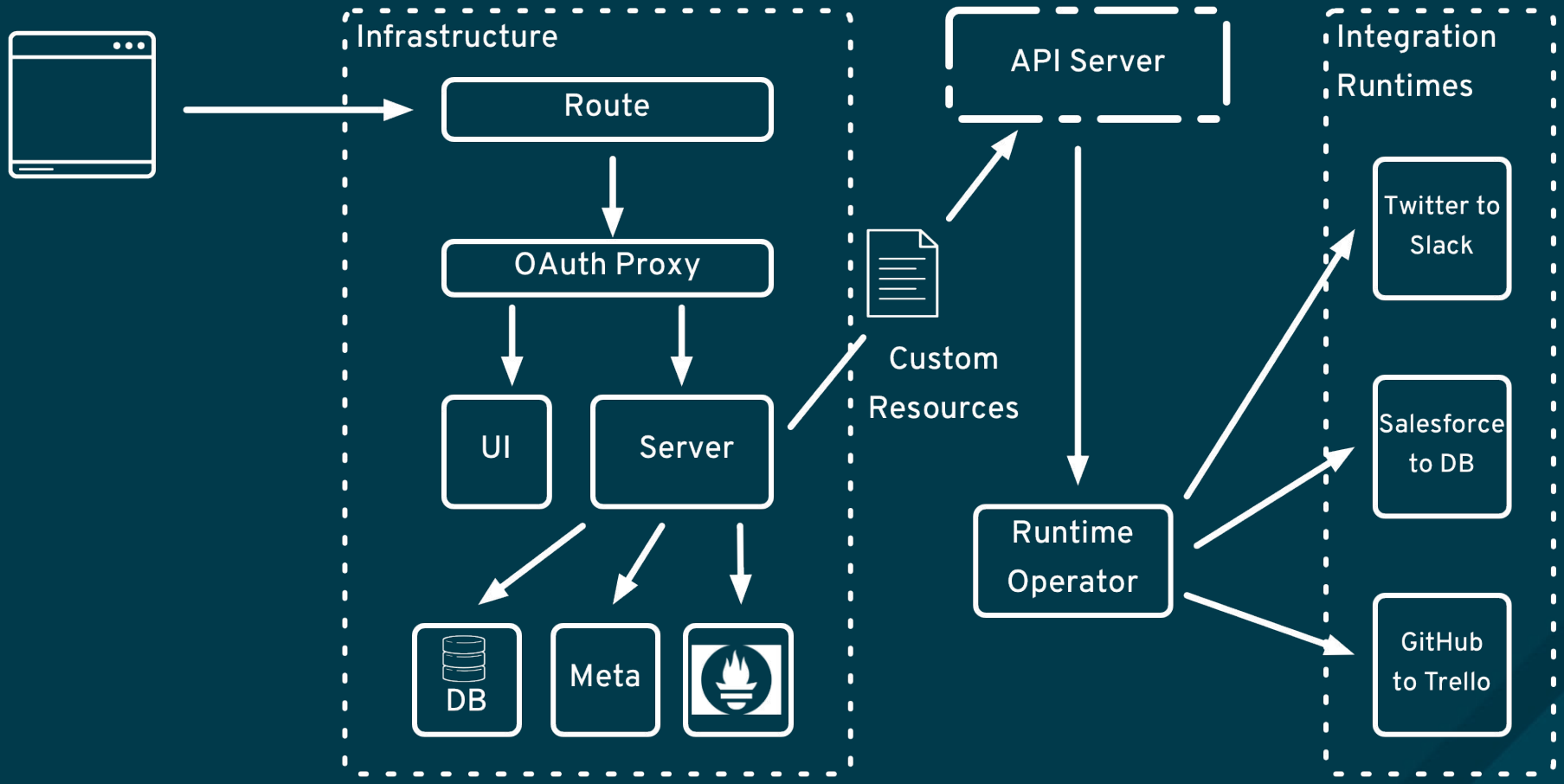
- Current Status:
  - Template based installation
  - Update Pod with a bunch of shell scripts
- Introduce Operator for Installation and Update
  - Instantiate the Template by the operator if not existent
  - Run update Pod if Syndesis is already installed
- Alternative: Ansible Playbook Bundles executed by the Ansible Service Broker
- Questions:
  - How is the operator itself installed ? Has it to be clusterwide ?
  - How is the error handling when update fails ?

CRD Syndesis

```
kind: Syndesis
apiVersion: syndesis.io/v1
metadata:
  name: syndesis
  version: 1.4.2
spec:
  routeHostName: ${project}.6a63.fuse-ignite.openshiftapps.com
  demoData: false
  maxIntegrations: 5
  postgresql:
    memoryLimit: 255Mi
    user: syndesis
  prometheus:
    memoryLimit: 512Mi
    volumeSize: 1Gi
  server:
    memoryLimit: 800Mi
  meta:
    memoryLimit: 512Mi
  ...
```

RUNTIME OPERATOR

RUNTIME OPERATOR - STAGE 1



RUNTIME OPERATOR - STAGE 1

Replace Syndesis internal Controller

- Introduce CRDs `Integration` (and `Connection` ?)
- Remove Syndesis internal Controller
- On Publish Action:
  - Create `Integration` resources describing the runtime
  - Runtime Operator performs:
    - Create S2I Build
    - Create Deployment
- Runtime Operator updates `Integration` status based on Build/Deployment status
- Server listens to `Integration` changes and updates DB/UI accordingly

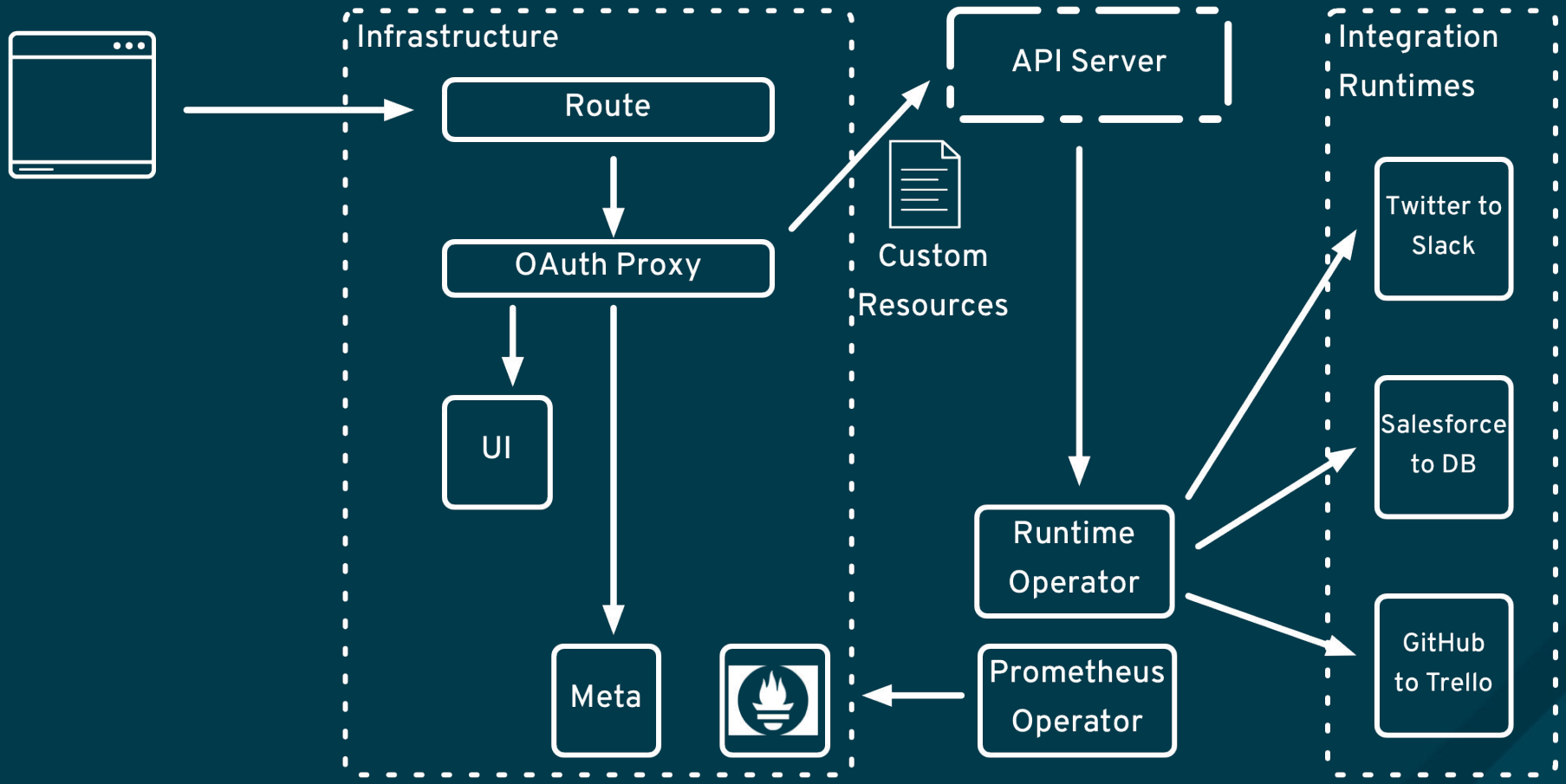
CRD Integration

```
kind: Integration
apiVersion: syndesis.io/v1
metadata:
  name: twitter-to-salesforce
  tags: [ "twitter", "salesforce" ]
spec:
  steps:
  - connection:
      name: twitter-search-kubernetes-patterns
  - mapping:
      valueFrom:
        configMapKeyRef:
          name: mappings-config
          key: twitter-to-salesforce
  - connection:
      name: salesforce-create-or-update-contact
```

CRD Connection

```
kind: Connection
apiVersion: syndesis.io/v1
metadata:
  name: twitter-search-kubernetes-patterns
spec:
  action: io.syndesis:twitter-search-action
  outputShape:
    kind: java
    class: twitter4j.Status
  properties:
  - name: keywords
    value: "syndesis"
  - name: interval
    value: "5000"
  - name: accessToken
    value: 4593e95fa67ec90b0012345aa345
  - name: accessTokenSecret
    valueFrom:
      secretKeyRef:
        name: twitter-secret
        key: accessTokenSecret
```

RUNTIME OPERATOR - STAGE 2



RUNTIME OPERATOR - STAGE 2

Replace Syndesis Server Component

- Remove state handling from Syndesis by using custom resources exclusively
- UI (Angular 5 app) accesses OpenShift API server directly to manage custom resources
  - Massive change required, not sure if this even possible
- Benefits:
  - No Database to maintain
  - No API to define
  - Reduces resource footprint (no syndesis-server, no db, one PV less)
  - Multitenancy
  - HA
- Use Prometheus Operator
  - `ServiceMonitor` objects are either pre-created or on the fly by the Syndesis runtime operator