Open Application Model

Kubernetes has provided a useful set of APIs to orchestrate container primitives

But how do we stitch these into an operable application?

DevOps + Orchestrators

Large ratio app dev to infra/app ops

Homemade PaaS/FaaS built to abstract orchestrator

Overly complex CI/CD pipelines

Open App Model

Platform-agnostic open source specification that defines cloud native applications.

Designed to solve how distributed apps are composed and transferred to those responsible for operating them

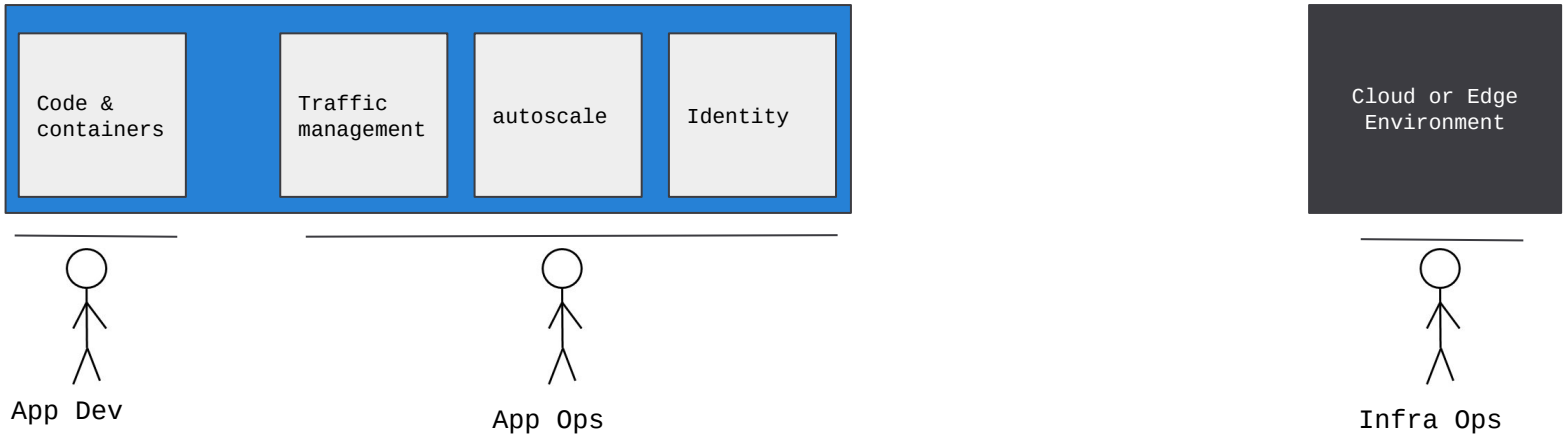
Currently in prerelease v1.0.0.alpha1

OAM Principals

- Application focused — **Focuses** on developers and **applications**, not on container infrastructure
- Separation of concerns — **Clearly defined roles** for application developers, application operators, and infrastructure operators
- Cloud + Edge — **Consistent** application **modeling** for cloud, on-prem, and small-edge devices

OAM Personas

- Allows **application developers** to focus on their code in a platform-neutral setting to deliver business value
- **Application operators** use powerful and extensible operational traits consistently across platforms and environments
- **Infrastructure operators** can configure their environments to satisfy any unique operating requirements



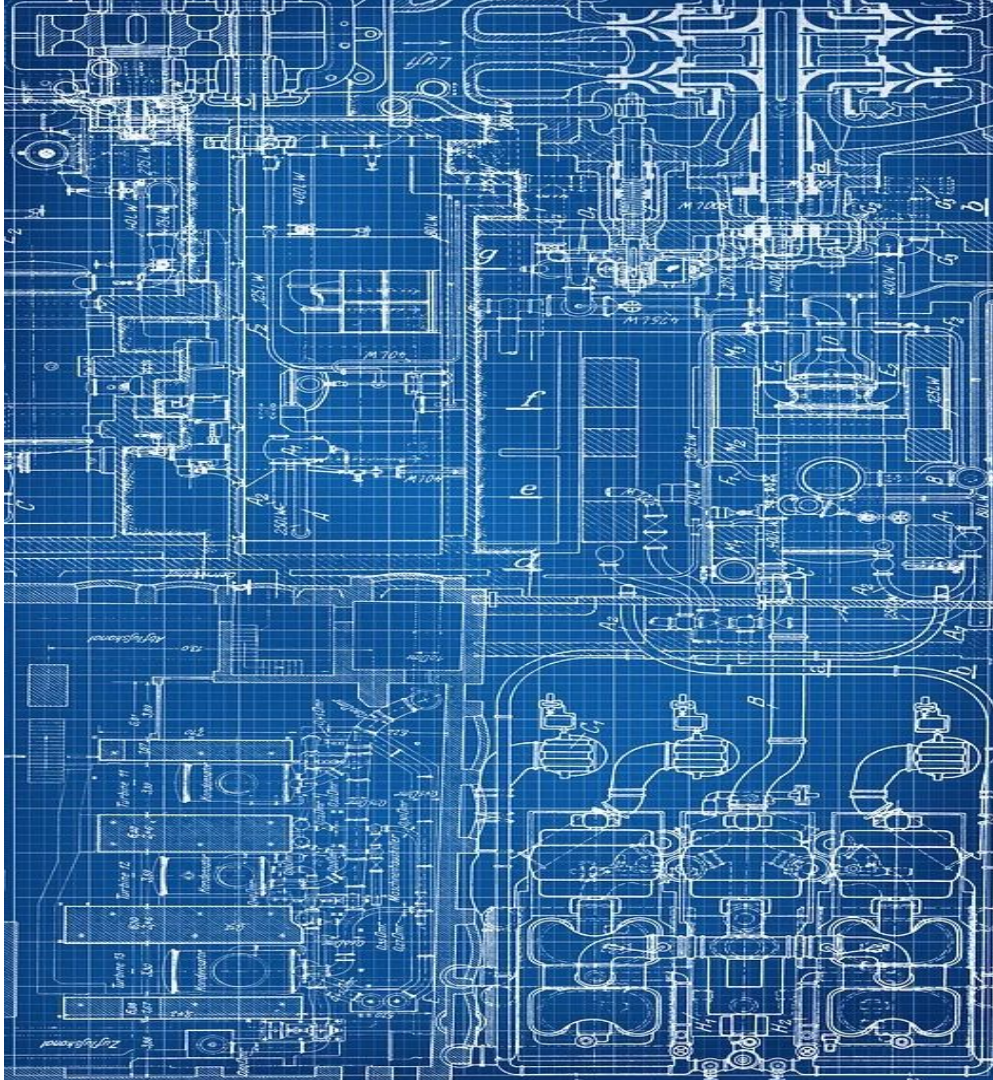
OAM Constructs

Components

Traits

Application Scopes

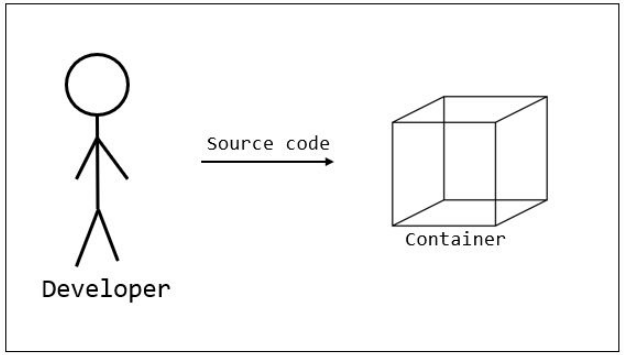
Application Configuration

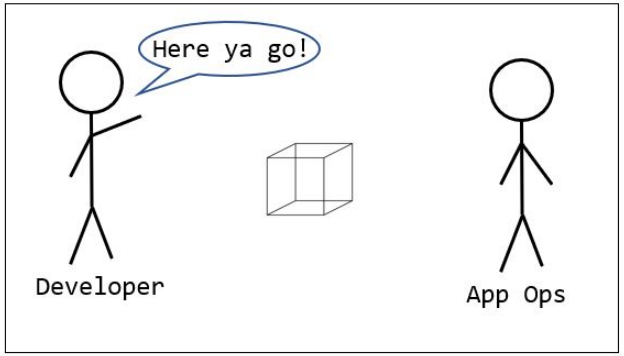


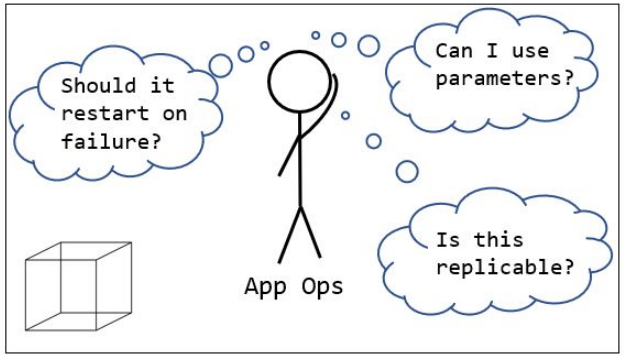
Components

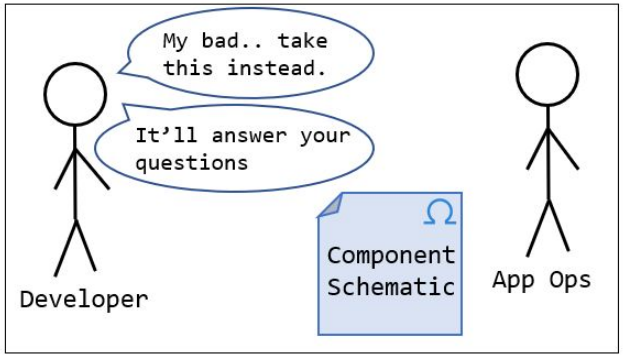
Purpose: Encapsulate application code

- Workload-type
- Parameters
- Resource Requirements
- Health/liveliness probes





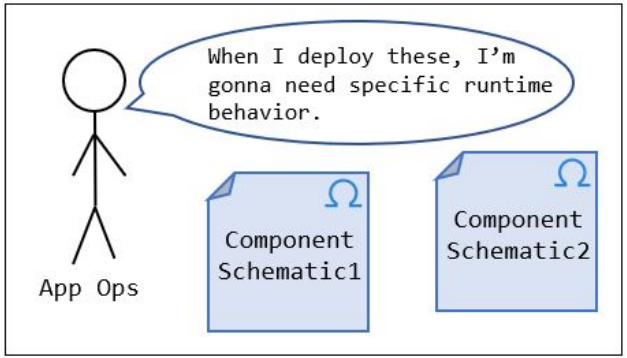


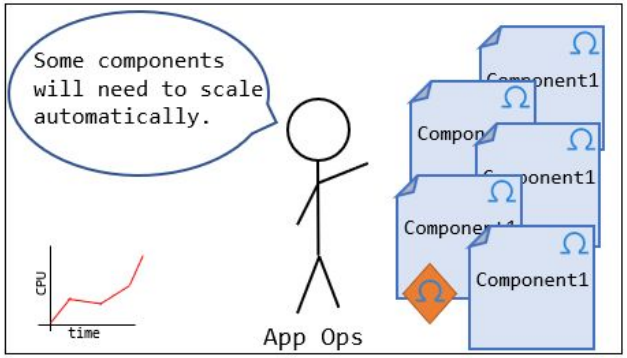


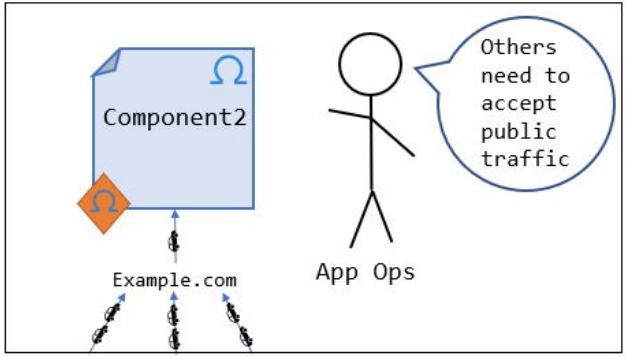
Traits

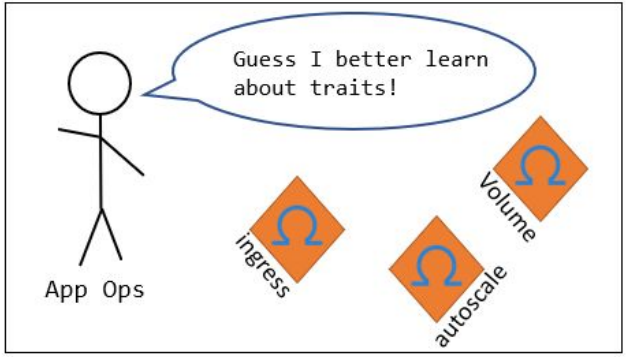
Purpose: Discretionary runtime overlays

- Operational functionality to component instances





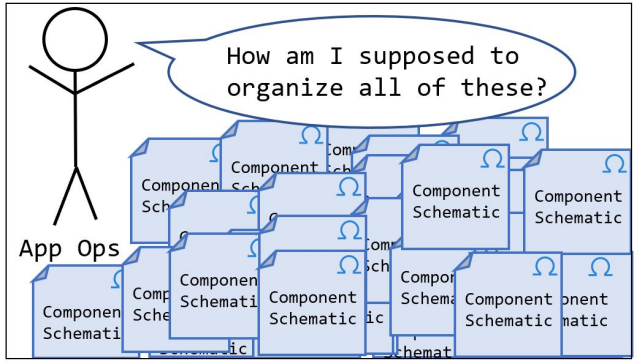


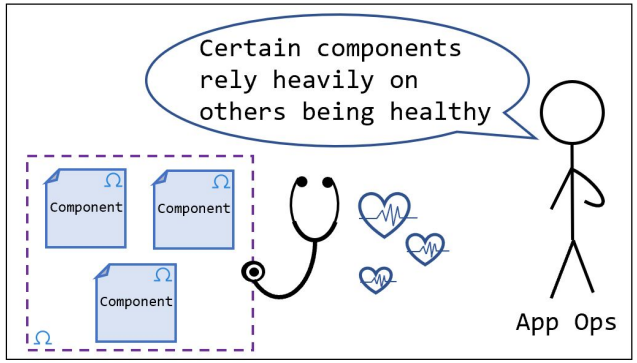


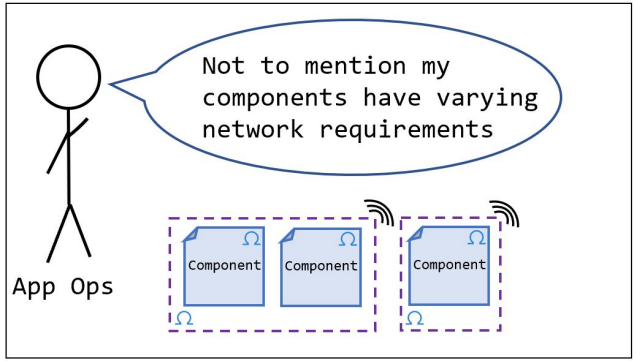
Application Scopes

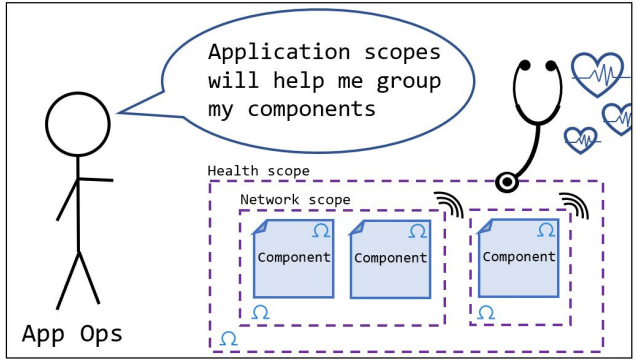
Purpose: Discretionary application boundaries

- Group behaviors for components





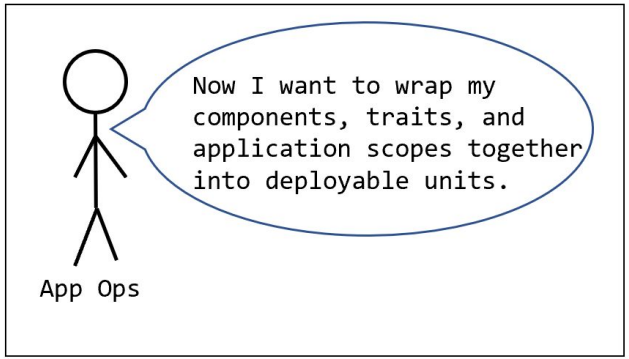


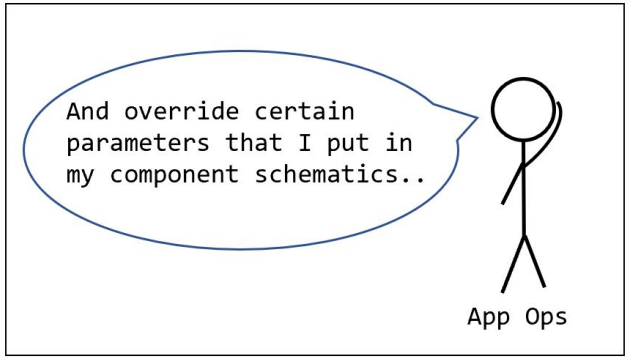


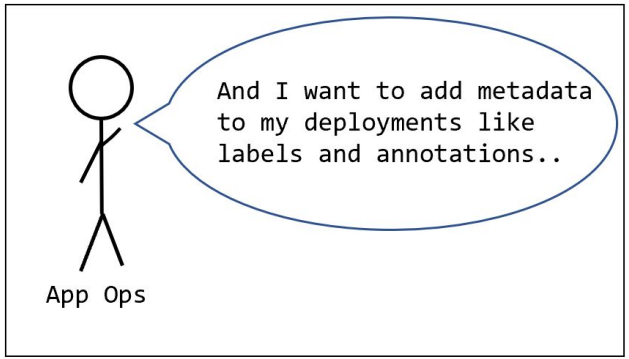
Application Configurations

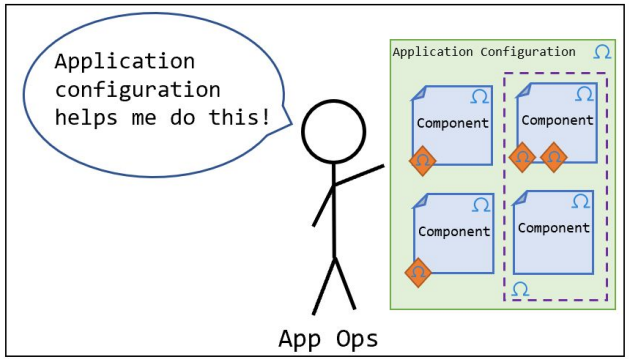
Purpose: Defines application deployment

- Deploy components, traits, and application scopes

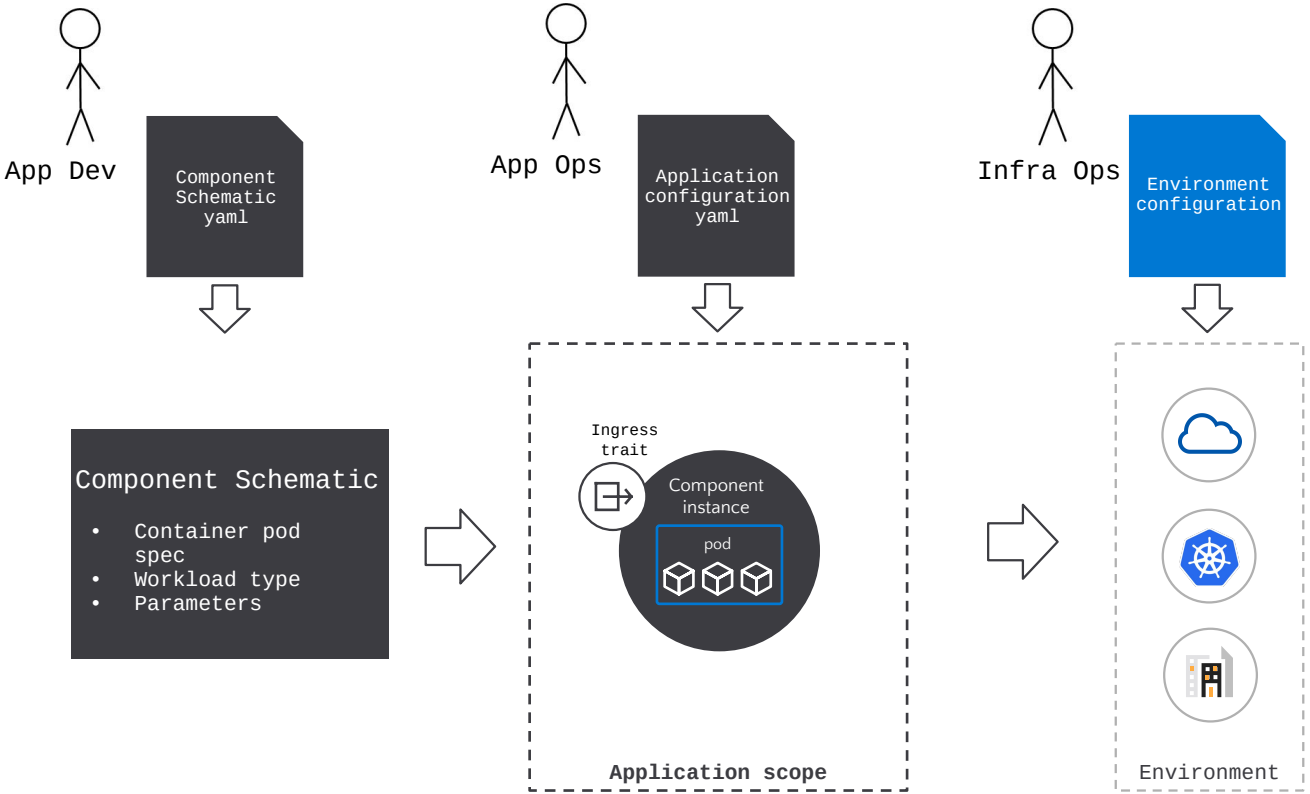




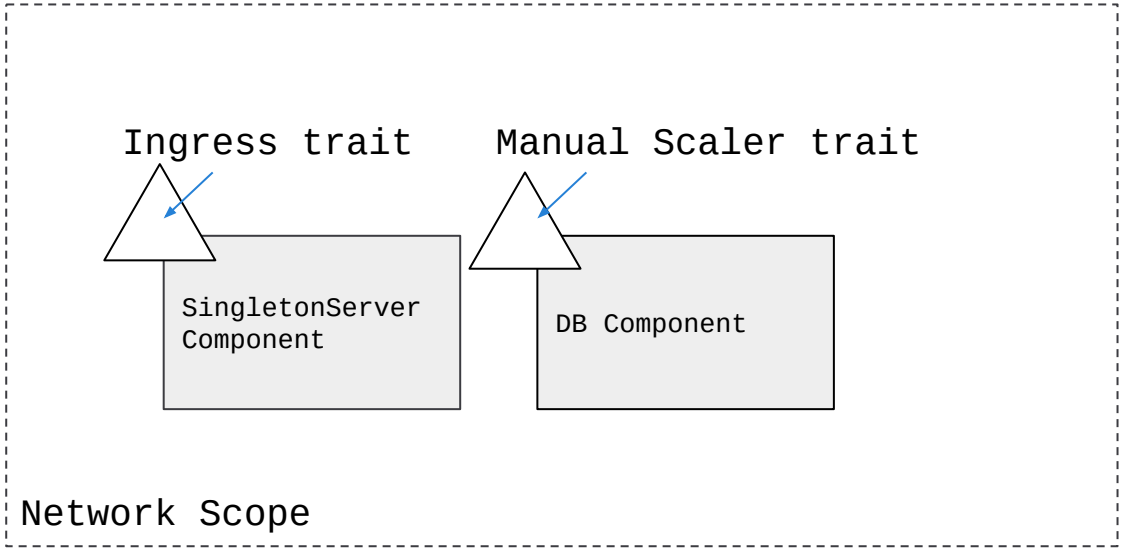




Putting it together



Simple Example



Component Schematics

```
apiVersion: core.oam.dev/v1alpha1
kind: ComponentSchematic
metadata:
  name: web-ui
spec:
  workloadType: core.oam.dev/v1.SingletonServer
  osType: Linux
  parameters:
     - name: DB_SECRET
       type: string
       required: true
  containers:
   - name: web-server
     image: example/web-server:v1
       resources:
          cpu: 1
          mem: 200MB
     ports:
     - name: webport
       value: 8080
       Protocol: TCP
     env:
     - name: dbSecret
       fromParam: DB_SECRET
```

```
apiVersion: core.oam.dev/v1alpha1
kind: ComponentSchematic
metadata:
  name: mongo-db
spec:
  workloadType:core.oam.dev/v1.Server
  osType: Linux
  containers:
   - name: mongodb
     image: docker.io/bitnami/mongodb:4.0.10-debian-9-r39
       resources:
          cpu: 2
          mem: 600MB
     ports:
     - name: dbport
       value: 27017
       protocol: TCP
```

OAM Core Workload Types

Name	Type	Service endpoint	Replicable	Daemonized
Server	core.oam.dev/v1alpha1.Server	Yes	Yes	Yes
Singleton Server	core.oam.dev/v1alpha1.SingletonServer	Yes	No	Yes
Worker	core.oam.dev/v1alpha1.Worker	No	Yes	Yes
Singleton Worker	core.oam.dev/v1alpha1.SingletonWorker	No	No	Yes
Task	core.oam.dev/v1alpha1.Task	No	Yes	No
Singleton Task	core.oam.dev/v1alpha1.SingletonTask	No	No	No

Trait and Scope

```
apiVersion: core.oam.dev/v1alpha1
kind: Trait
metadata:
  name: manual-scaler
  annotations:
    version: v1.0.0
    description: "Allow operators to manually scale a workloads
that allow multiple replicas."
spec:
  appliesTo:
    - core.oam.dev/v1alpha1.Server
    - core.oam.dev/v1alpha1.Worker
    - core.oam.dev/v1alpha1.Task
  Properties:
     - name: replicaCount
       Description: The number of instances required to be running
       Required: Y
       Type: integer
```

```
apiVersion: core.oam.dev/v1alpha1
kind: ApplicationScope
metadata:
  name: network
  annotations:
    version: v1.0.0
    description: "network boundary that a group components res
in"
spec:
  type: core.oam.dev/v1.NetworkScope
  allowComponentOverlap: false
  parameters:
    - name: network-id
      description: The id of the network, e.g. vpc-id, VNet na
      type: string
      required: Y
    - name: subnet-ids
      description: >
        A comma separated list of IDs of the subnets within th
network. For example, "vsw-123" or ""vsw-123,vsw-456".
        There could be more than one subnet because there is a
limit in the number of IPs in a subnet.
        If IPs are taken up, operators need to add another sub
into this network.
      type: string
      required: Y
    - name: internet-gateway-type
      description: The type of the gateway, options are 'publi
'nat'. Empty string means no gateway.
      type: string
        required: N
```

Component Schematics → App Config

```
apiVersion: core.oam.dev/v1alpha1
kind: Component
metadata:
  name: web-ui
spec:
  ...
---
apiVersion: core.oam.dev/v1alpha1
kind: Component
metadata:
  name: mongo-db
spec:
  ...
```

```
apiVersion: core.oam.dev/v1alpha1
kind: ApplicationConfiguration
metadata:
  name: service-tracker
spec:
  scopes:
  - name: network
    type: core.oam/dev/v1alpha1.Network
     properties:
       network-id: “mynetworkID”
       subnet-ids: "subnetID1"
       internet-gateway-type: “public”
  components:
  - componentName: mongo-db
    instanceName: mongo-db
    traits:
    - name: manualScaler
      properties:
        replicaCount: 3
    applicationScopes:
    - network
  - componentName: web-ui
    instanceName: web-ui
    parameterValues:
    - name: DB_SECRET
      value: "supersecureconnectionstring"
    traits:
    - name: ingress
      properties:
         hostname: servicetracker.oam.io
         path:  /
         service_port: 8080
    applicationScopes:
     - network
```

How can I use the specification in practice?

Existing OAM implementations

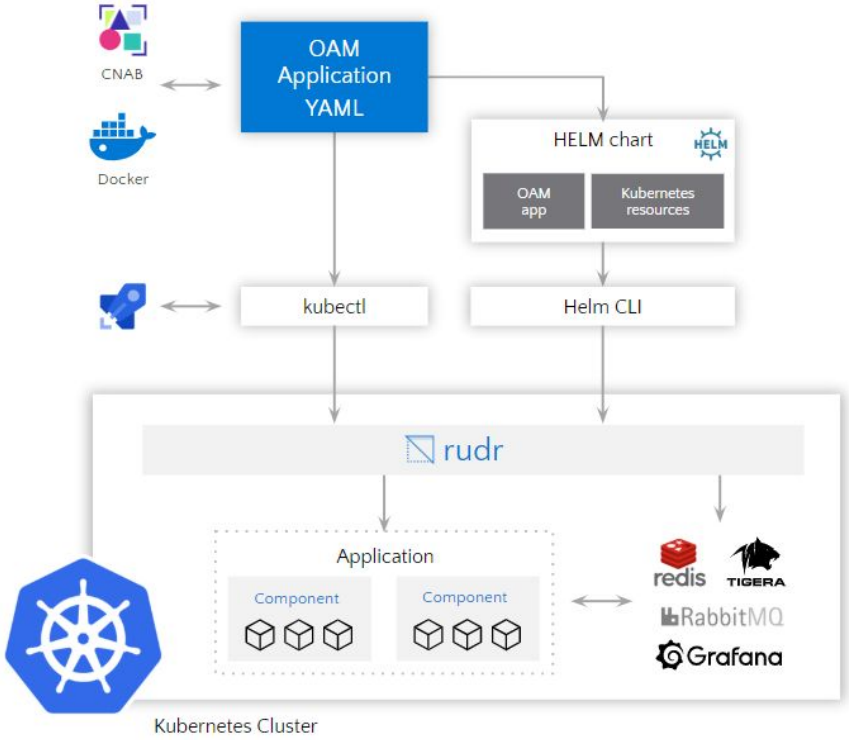
- Rudr
- Alibaba Enterprise Distributed Application Service
- Alibaba Resource Orchestration Service

Rudr - k8s reference implementation

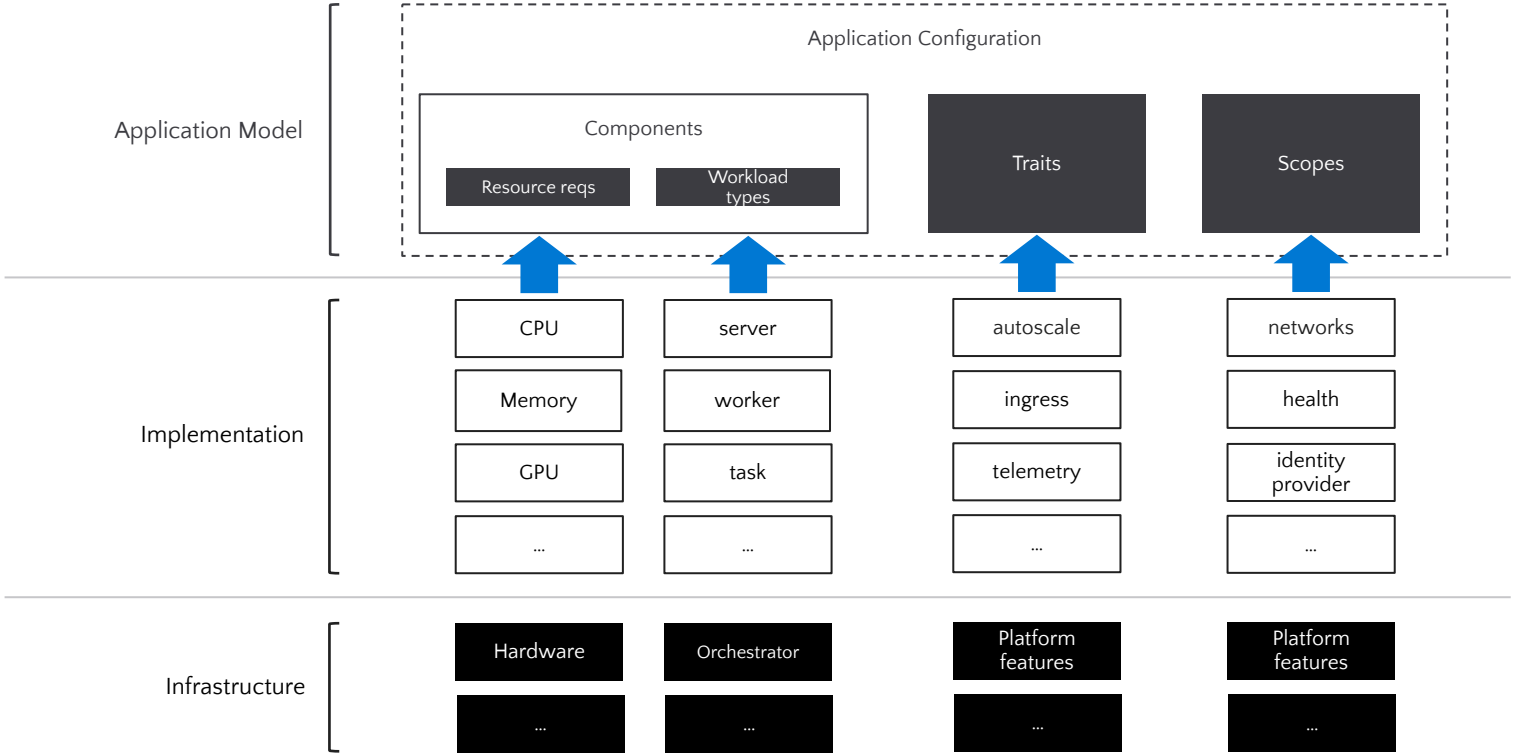
OSS project

Works on any k8s cluster

Supports all core OAM constructs



OAM Implementation Architecture



Hot Topics

OAM and Extensibility

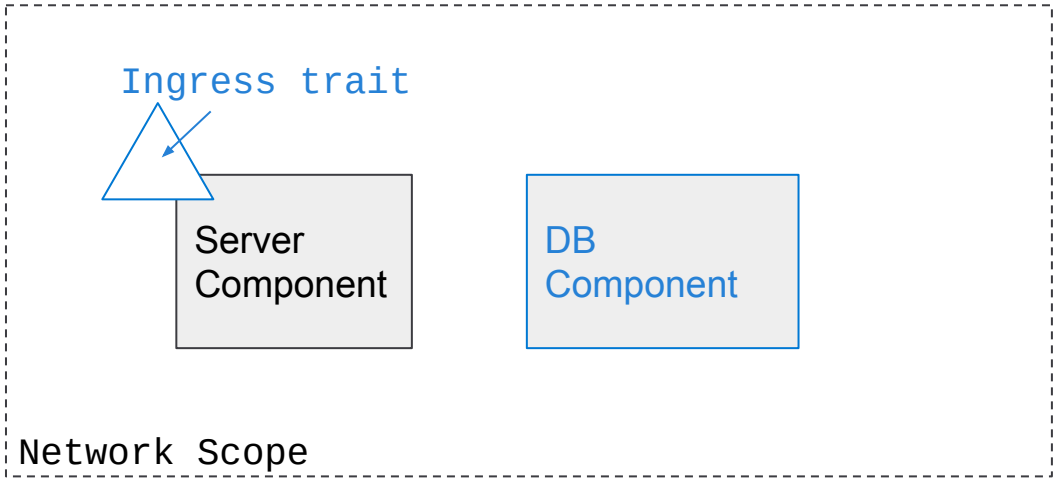
Today OAM supports two categories for component workload types, trait types, and application scope types

1. Core: **must be** supported by OAM compliant implementations
2. Extended: **optional** support by OAM compliant implementations

Core Construct Types in OAM

Component Workload Types	App Scope Types	Trait Types
Server	Network	Manual Scaler
Singleton Server	Health	
Job		
Singleton Job		
Task		
Singleton Task		

Example



Core = black
Extension = blue

Up and coming

OAM is not well positioned for infra operators interested in implementing extended workload types, extension traits/scopes

Second draft is focused on making OAM more flexible for infra ops by including existing resources into an OAM runtime

Community

Get Involved

Join the discussion

Join the Community Call (next call 2/25 @ 10:30am PST)

Subscribe to OAM Community Calendar

Contribute to repos: OAM Spec, Rudr

Application Scopes

Scope definition

Defined by an OAM implementation

Follows a standard schema for discoverability

Parameters for configuration

```
apiVersion: core.oam.dev/v1alpha1
kind: ApplicationScope
metadata:
  name: network
  annotations:
    version: v1.0.0
    description: "network boundary for components"
spec:
  type: core.oam.dev/v1.NetworkScope
  allowComponentOverlap: false
  parameters:
    - name: network-id
      description: The id of the network, e.g. vpc-id, VNet name.
      type: string
      required: yes
```

Traits

Trait definition

Defined by an OAM implementation

Follows a standard schema for discoverability

Properties JSON schema defines the trait's configuration options

```
apiVersion: core.oam.dev/v1alpha1
kind: Trait
metadata:
  name: ManualScaler
  annotations:
    version: v1.0.0
  spec:
  appliesTo:
    - core.oam.dev/v1alpha1.Server
    - core.oam.dev/v1alpha1.Worker
    - core.oam.dev/v1alpha1.Task
  properties:
    type: object
    properties: |
      {"$schema": "http://json-schema.org/draft-07/schema#",
        "type": "object",
        "required": ["replicaCount],
        "properties": {
          "replicaCount": {
            "type": "integer",
             "minimum": 0 }}}
```