KubePattern

Presenter: Gabriele Groppo
Advisor: Proff. Francesca Arcelli Fontana

Università degli Studi di Milano Bicocca

Analysis Tool for Improving Cluster and Deployments Quality Using **Patterns**.

Agenda

Project

- Key Features
- Kubernetes Integration

Architecture

- Graph Based Approach
- Relationships
- Analysis Explained

Patterns

- Pattern Kubernetes
- Pattern As Code
- Use cases

Key Features

Detection

Identifies potential **smells** in Kubernetes clusters

Suggestions

Provides actionable improvement, through **Patterns**

Open Source

Freely available for all users, to **customize**.

Integration

Seamlessly fits into workflows, thanks to APIs & **Kubernetes CRD** output

KubePattern integration

Kubernetes Custom Resource Definition
K8sPattern

KubePattern uses graph to analyse interactions between K8s Resources.

```yaml
# HEALTH PROBE PATTERN
apiVersion: kubepattern.it/v1
kind: K8sPattern
metadata:
 name: health-probe-missing-a689742f
 namespace: pattern-analysis-ns
spec:
 apiVersion: kubepattern.it/v1
 confidence: HIGH
 description: "..."
 message: "..."
 name: health-probe-missing
 docLink: "..."
 resources:
 - name: javaspringboots-v0-0-27-
controller-84599b5f5f-97zb6
```

KubePattern integration

Kubernetes Custom Resource Definition
K8sPattern

KubePattern uses graph to analyse interactions between K8s Resources.

```
...
docLink: "..."
resources:
- name: javaspringboots-v0-0-27
namespace: cliente1
role: single-pod
uid: c021cbe0-a59e-4d45
scores:
- relationships: 1
severity: INFO
type: FOUNDATIONAL
```

KubePattern integration

Kubernetes Custom Resource Definition
K8sPattern

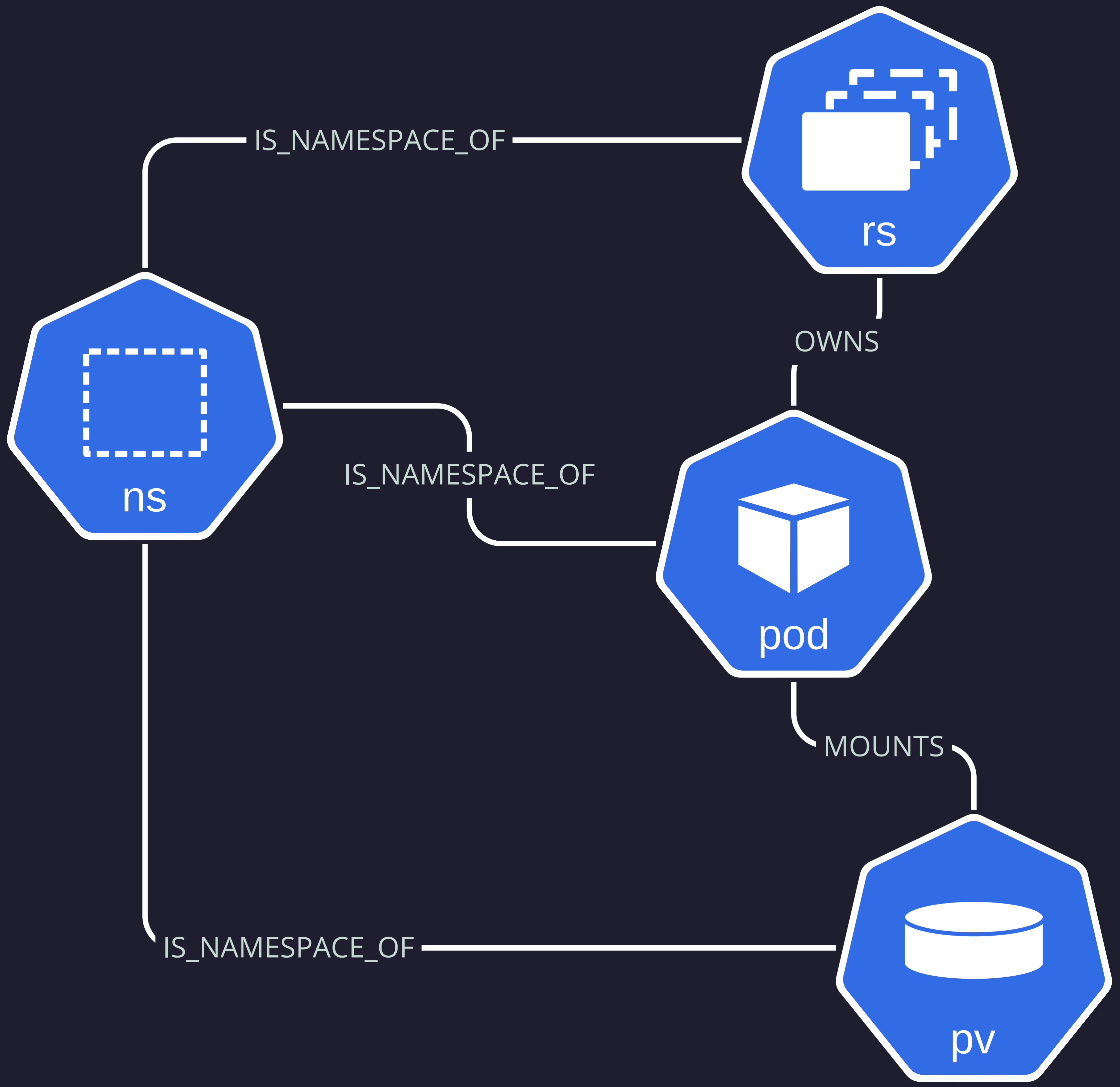
REST API's to Analyze Cluster, Lint Pattern as Code & Get Cluster Graph

```
GET /cluster/graph

POST /analysis/cluster

POST /analysis/cluster?pattern=sidecar

POST /analysis/namespace/{namespace}

POST /pattern/lint
```

Graph Based Approach

KubePattern uses graph to analyse interactions between K8s Resources.

This is the key to enforce complex Patterns.



Relationships

Strategy to assign relationships

Every strategy check for **matches in multiple yaml** files.

Each Strategy implements the same **Interface**: IRelationshipStrategy

VETO logic to filter resources that does not met **sharing requirements**.

There are **as many Strategies as Relationships**

Relationships

Strategy to assign relationships

Every strategy check for **matches in multiple yaml** files.

Each Strategy implements the same **Interface**: IRelationshipStrategy

VETO logic to filter resources that does not met **sharing requirements**.

With Strategy design pattern every relationship has same interface but **different concrete implementation**.

Relationships

Strategy to assign relationships

Every strategy check for **matches in multiple yaml** files.

Each Strategy implements the same **Interface**: IRelationshipStrategy

VETO logic to filter resources that does not met **sharing requirements**.

In order to establish relationships there is a match between fields in different yaml files for **target resource kinds**.

Relationships

Strategy to assign relationships

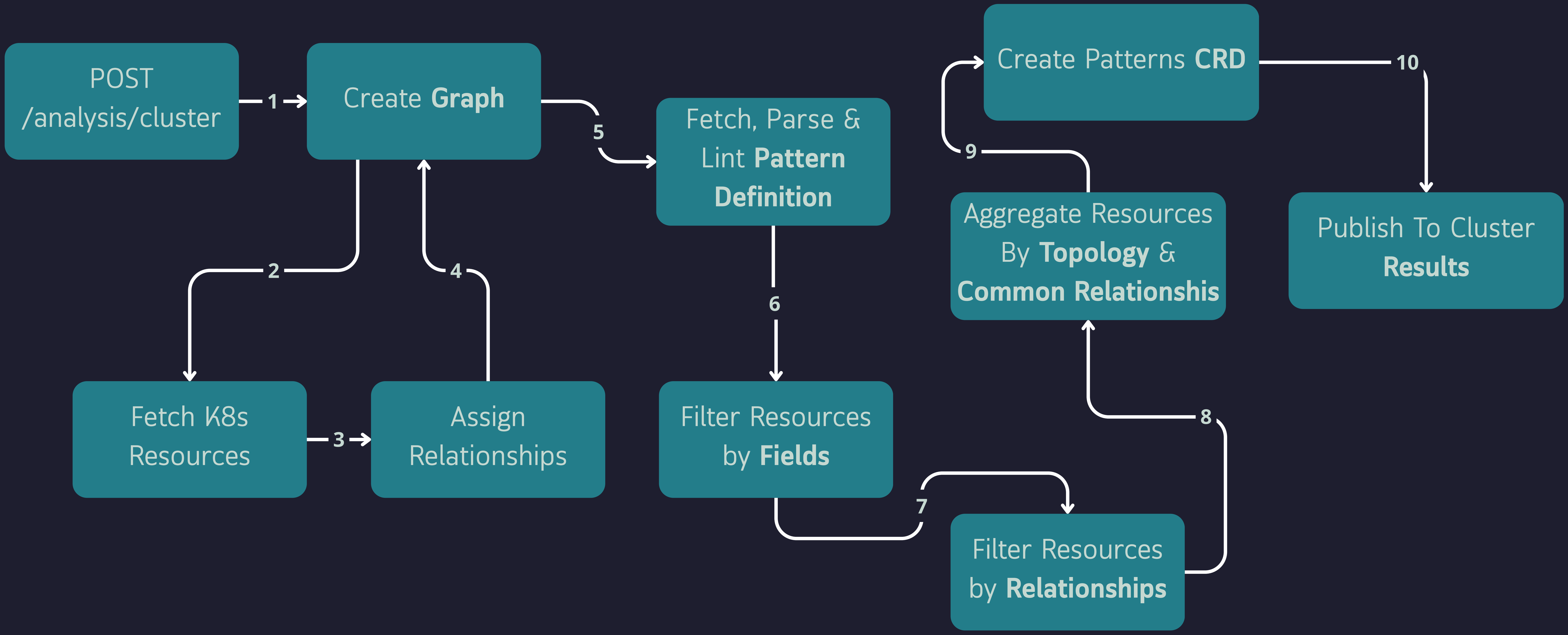
Every strategy check for **matches in multiple yaml** files.

Each Strategy implements the same **Interface**: IRelationshipStrategy

VETO logic to filter resources that does not met **sharing requirements**.

Relationships in pattern can or cannot be **required**. They also must or must not be **shared**.

KubePattern Analysis


POST /analysis/cluster
1
Create **Graph**
2
Fetch K8s Resources
3
Assign Relationships
4
5
Fetch, Parse & Lint **Pattern Definition**
6
Filter Resources by **Fields**
7
Filter Resources by **Relationships**
8
Aggregate Resources By **Topology** & **Common Relationshis**
9
Create Patterns **CRD**
10
Publish To Cluster **Results**

Pattern As Code

```yaml
version: kubepattern.dev/v1
kind: Pattern
metadata:
  name: sidecar # Unique name in registry
  displayName: Sidecar
  patternType: STRUCTURAL
  severity: INFO
  category: architecture
  gitUrl: https://github.com/kubepattern/registry//k8s-sidecar.json
  docUrl: https://github.com/kubepattern/registry//k8s-sidecar.md
spec:
  message: |
```

Pattern As Code

```
...
  docUrl: https://github.com/kubepattern/registry//k8s-sidecar.md
spec:
  message: |
    Pod '{{main-app.name}}' in namespace '{{main-app.namespace}}'
    appears to be separated from its sidecar pod '{{sidecar.name}}' in
    namespace '{{sidecar.namespace}}'.
  topology: LEADER_FOLLOWER
  resources:
  - resource: Pod
    id: main-app
    leader: true
```

Pattern As Code

```yaml
    namespace: "{{sidecar.namespace}}"
  topology: LEADER_FOLLOWER
  resources:
  - resource: Pod
    id: main-app
    leader: true # This is the leader resource (is unique in pattern)
    filters:
      matchAll: # All conditions must be met
      - key: ".spec.volumes" # Json field
        operator: EXISTS
        values: []
      matchNone: # None of the conditions must be met
      - key: ".metadata.namespace"
```

Pattern As Code

```yaml
  - resource: Pod
    id: sidecar
    filters:
      matchAll:
      - key: ".spec.volumes"
        operator: EXISTS
        values: []
      matchNone:
      - key: ".metadata.namespace"
        operator: EQUALS
        values:
        - kube-system
      matchAny:
```

Pattern As Code

```yaml
      values:
      - logging
actors:
- main-app
- sidecar
commonRelationships:
- id: shared-volume-mount
  type: MOUNTS
  required: true
  shared: true
  resourceIds:
  - main-app
  - sidecar
```

Pattern As Code

```
  - sidecar
  commonRelationships:
  - id: shared-volume-mount
    type: MOUNTS
    required: true
    shared: true
    resourceIds:
    - main-app
    - sidecar
  relationships:
  - id: my-rel-1
    type: REL_1
    weight: 1
```

Pattern As Code

```yaml
relationships:
- id: my-rel-1
  type: RELATIONSHIP_TYPE
  weight: 1 # Points to sum if matched if not required
  required: true # Must be matched
  shared: false # Must or must not be shared
  resourceIds: #Resources to match
  - main-app
  - sidecar
minRelationshipPoints: 1 #Points required to pass relationship Filter
```

Pattern Kubernetes

Kubernetes Patterns are **reusable architectural templates** and **best practices** for designing, building, and maintaining cloud-native applications on Kubernetes.

K8sPatterns

FOUNDATIONAL BEHAVIORAL STRUCTURAL

ADVANCED CONFIGURATION SECURITY CUSTOM

Pattern Kubernetes

FOUNDATIONAL

- health probe
- predictable demands
- automated placement

BEHAVIORAL

- batch job
- stateful service
- stateful discovery

ADVANCED

- git ops
- operator

STRUCTURAL

- sidecar
- init container

Selected Use cases

Compliance

Ensure **policy & best practices** across clusters.

Reliability

Improve **configuration stability** & automate **failure recovery**.

Custom Resources

Removed **zombie resources** from cluster to improve efficiency.

Structure

Improve cluster resources **life cycle** & **interactions**.

Questions? Reach out for more information

Email

g.groppo@campus.unimib.it

Official KubePattern Website

kubepattern.dev