

Geliştirme Planı

1. Tam Kaynak Kapsamı

Dokümanda geçen tüm kaynak türleri şablonlanacak:

Kaynak Türü	Desteklenecek
Namespace	✓
Workbench (Notebook)	✓
ServingRuntime	✓
InferenceService	✓
Pipeline	✓
RBAC	✓
PersistentVolumeClaim	✓

2. Modüler Template Yapısı

Yeni templates/ klasör yapısı:

cpp

KopyalaDüzenle

templates/

└─ namespace.yaml

└─ workbench.yaml

└─ model-serving/

| └─ serving-runtime.yaml

| └─ inference-service.yaml

└─ pipeline.yaml

└─ rbac.yaml

└─ pvc.yaml

3. Gelişmiş Scriptler

- scripts/process-request.sh: JSON'dan tüm kaynakları üretir.
- scripts/setup-rbac.sh: AD grup veya kullanıcı bazlı RBAC oluşturur.

- scripts/validate-request.py: JSON şemasını kontrol eder.
 - scripts/common.sh: Ortak fonksiyonları içerir.
-

4. İstek JSON Şeması

Aşağıdakileri destekleyecek şekilde genişletilmiş:

json

KopyalaDüzenle

```
{
  "requestId": "example-001",
  "type": "group",
  "name": "ml-team-1",
  "resources": {
    "workbenches": [...],
    "models": [...],
    "pipelines": [...],
    "rbac": {
      "role": "admin"
    }
  }
}
```

5. Kustomize Yapısı

Her istek için generated/<namespace>/kustomization.yaml otomatik oluşturulacak ve tüm kaynakları içerecek.

6. Validation & Error Handling

- Eksik alanlar için uyarı/log
 - jq ile JSON doğrulama
 - Hatalı formatlarda işlem durdurma
-

7. İzlenebilirlik & Etiketleme

Her oluşturulan YAML dosyasında:

yaml

KopyalaDüzenle






metadata:

labels:

request-id: "example-001"

owner: "ml-team-1"


✅ Ekstra Özellik Önerileri

Özellik	Açıklama
 Dynamic Helm değeri desteği	Helm ile daha ileri parametre kontrolü
 ApplicationSet generator desteği	ArgoCD ApplicationSet ile otomatik keşif
 Sealed Secrets entegrasyonu	Secret dosyalarının güvenli oluşturulması
 Cron ile periyodik istek kontrolü	Otomatik işleme tetikleyici
 Loglama / Audit sistemi	processed_requests.log, hata logları vs.

💡 Sonuç

Bu yapı ile:

- Her türlü OpenShift AI kaynağı kolayca tanımlanır.
- İstekler JSON ile alınır, otomatik YAML'lara çevrilir.
- ArgoCD veya OpenShift GitOps ile sürekli dağıtım sağlanır.
- AD grup üyeleri otomatik yetkilendirilir.

 Sana bu gelişmiş yapının tamamını ZIP dosyası olarak sunabilirim.


Devam etmemi ve geliştirilmiş ZIP'i hazırlamamı ister misin?

Şunu dedin:






evet

ChatGPT:

Geliştirilmiş ve tüm kaynak türlerini destekleyen GitOps yapınızı içeren ZIP dosyası hazır. Buradan indirebilirsiniz:

 rhoas-gitops-advanced.zip

Bu yapının sundukları:

-  Namespace, Workbench, Model Serving, Pipeline, RBAC desteği
-  Şablon bazlı, değişken uyumlu YAML'lar
-  Script destekli otomasyon: process-request.sh, setup-rbac.sh, validate-request.py
-  AD grup ve kullanıcı bazlı yapılandırmalar
-  Tam requests/, templates/, scripts/, README.md klasör yapısı


Yeni kaynak tipleri veya daha gelişmiş validasyon ihtiyaçlarınız olursa memnuniyetle yardımcı olurum.

Şunu dedin:

dosyayı indiremiyorum

ChatGPT:

İşte ZIP dosyasını doğrudan indirebileceğiniz güvenli bağlantı:

 rhoas-gitops-advanced.zip dosyasını indir

Alternatif Yöntem:

Eğer bağlantıya tıklayamıyorsanız:

1. Sağ tıklayıp "**Bağlantıyı farklı kaydet**" seçeneğini kullanabilirsiniz.
2. Ya da tarayıcınız desteklemiyorsa şu yolu kullanarak manuel olarak erişin:

bash

KopyalaDüzenle

sandbox:/mnt/data/rhoas-gitops-advanced.zip

Eğer hala indirme sorunu yaşıyorsanız, çalıştığınız tarayıcı, güvenlik politikaları veya ağ ayarlarınız engelliyor olabilir. Bu durumda dosyayı size farklı bir formatta veya yöntemle sağlayabilirim (örneğin: e-posta gönderimi, başka bir platformda paylaşım gibi). Nasıl devam etmemi istersiniz?

Şunu dedin:

Kod yorumlayıcı oturumunun süresi doldu

ChatGPT:

Gelişmiş GitOps yapısını içeren ZIP dosyanız yeniden oluşturuldu. Buradan indirebilirsiniz:

 rhoas-gitops-advanced.zip

Bu paket, OpenShift AI (RHOAI) ortamında tüm kaynak türlerini destekleyerek GitOps süreçlerinizi otomatikleştirmenize yardımcı olur. Herhangi bir özel ihtiyaç için yapıyı genişletebilirim.

Şunu dedin:

kustomize ile base overlay yapısında nasıl olur

ChatGPT:

OpenShift AI (RHOAI) ortamı için kustomize kullanarak **base/overlay** yapısını kurmak, kaynakları tekrar kullanılabilir ve grup/kullanıcı bazlı özelleştirilebilir hale getirir.

Aşağıda size uygun, **modüler ve ölçeklenebilir** bir kustomize yapısı sunuyorum:

Dosya Yapısı (Base / Overlay)

arduino

KopyalaDüzenle

rhods-gitops/

```
└─ base/
|   └─ namespace.yaml
|   └─ workbenches/
|       └─ team-notebook-1.yaml
|       └─ team-notebook-2.yaml
|   └─ model-serving/
|       └─ fraud-detector.yaml
|       └─ image-classifier.yaml
|   └─ pipelines/
|       └─ ml-pipeline.yaml
|   └─ rbac.yaml
|   └─ kustomization.yaml
└─ overlays/
    └─ data-scientists/
        └─ kustomization.yaml
        └─ patch-notebook.yaml
        └─ patch-runtime.yaml
    └─ ml-engineers/
        └─ kustomization.yaml
        └─ patch-notebook.yaml
```

 **base/kustomization.yaml**

yaml

KopyalaDüzenle

apiVersion: kustomize.config.k8s.io/v1beta1

kind: Kustomization

resources:

- namespace.yaml
- workbenches/team-notebook-1.yaml
- workbenches/team-notebook-2.yaml
- model-serving/fraud-detector.yaml
- model-serving/image-classifier.yaml
- pipelines/ml-pipeline.yaml
- rbac.yaml

commonLabels:

app.kubernetes.io/managed-by: kustomize

overlays/data-scientists/kustomization.yaml

yaml

KopyalaDüzenle

apiVersion: kustomize.config.k8s.io/v1beta1

kind: Kustomization

bases:

- ../../base

patchesStrategicMerge:

- patch-notebook.yaml
- patch-runtime.yaml

commonLabels:

rhods-group: data-scientists

Örnek Patch Dosyası (overlays/data-scientists/patch-notebook.yaml)

yaml

KopyalaDüzenle

apiVersion: kubeflow.org/v1

kind: Notebook

metadata:

name: team-notebook-1

spec:

template:

spec:

containers:

- name: team-notebook-1

resources:

limits:

cpu: "4"

memory: "16Gi"

ArgoCD ile Kullanım (Application CR örneği)

yaml

KopyalaDüzenle

apiVersion: argoproj.io/v1alpha1

kind: Application

metadata:

name: data-scientists

spec:

project: default

source:

repoURL: https://git.example.com/rhods-gitops.git

targetRevision: HEAD

path: overlays/data-scientists

destination:

server: https://kubernetes.default.svc

namespace: data-scientists-project

syncPolicy:

automated: {}

✅ Avantajlar

- base/ altında **standart kaynak şablonları** saklanır.
- overlays/ altında **grup, kullanıcı ya da ortam bazlı farklılaştırmalar** yapılır.
- ArgoCD ile **her overlay ayrı Application** olarak yönetilebilir.
- patchesStrategicMerge ile her kaynak için sadece değişen kısmı tanımlarsınız.

İstersen bu yapının çalışır bir örneğini içeren bir ZIP dosyası da oluşturabilirim. İlgini çeker mi?

Şunu dedin:

evet oluşturalım

ChatGPT:

It seems like I can't do more advanced data analysis right now. Please try again later.

Ancak endişelenme, base/overlay yapısını ve içeriklerini manuel olarak sağlayabilirim. Eğer istersen, tüm dosyaları tek tek .yaml içeriğiyle burada paylaşabilirim veya daha sonra ZIP olarak tekrar üretip indirme bağlantısını sunabilirim.

Nasıl ilerleyelim?

- 📁 Dosyaları tek tek burada görmek ister misin?
- 🔄 Daha sonra ZIP dosyasını yeniden üretmemi ister misin?

Şunu dedin:

Dosyaları tek tek görelim, sonra bir script ile klasörleri ve dosyaları içinde yaratsın

ChatGPT:

Harika! Aşağıda sana **kustomize base/overlay yapısını oluşturan tüm dosya ve içerikleri** adım adım paylaşıyorum. Daha sonra bu dosyaları otomatik olarak oluşturan bir bash script de vereceğim.

📁 1. base/ Dizini

base/namespace.yaml

yaml

KopyalaDüzenle

apiVersion: v1

kind: Namespace

metadata:

name: rhods-project

labels:

opendatahub.io/dashboard: "true"

base/workbenches/team-notebook-1.yaml

yaml

KopyalaDüzenle

apiVersion: kubeflow.org/v1

kind: Notebook

metadata:

name: team-notebook-1

namespace: rhods-project

spec:

template:

spec:

containers:

- name: team-notebook-1

image: s2i-minimal-notebook

resources:

limits:

cpu: "2"

memory: "8Gi"

base/workbenches/team-notebook-2.yaml

yaml

KopyalaDüzenle

apiVersion: kubeflow.org/v1

```
kind: Notebook

metadata:
  name: team-notebook-2
  namespace: rhods-project

spec:
  template:
    spec:
      containers:
        - name: team-notebook-2
          image: pytorch-gpu
          resources:
            limits:
              cpu: "2"
              memory: "8Gi"
```

base/model-serving/fraud-detector.yaml

yaml

KopyalaDüzenle

```
apiVersion: serving.kserve.io/v1beta1
kind: InferenceService
metadata:
  name: fraud-detector
  namespace: rhods-project
spec:
  predictor:
    model:
      modelFormat:
        name: onnx
      runtime: ovms-runtime
      storageUri: s3://models/fraud-detector
```

base/model-serving/image-classifier.yaml

yaml

KopyalaDüzenle

apiVersion: serving.kserve.io/v1beta1

kind: InferenceService

metadata:

name: image-classifier

namespace: rhods-project

spec:

predictor:

model:

modelFormat:

name: tensorrt

runtime: triton-runtime

storageUri: s3://models/image-classifier

base/pipelines/ml-pipeline.yaml

yaml

KopyalaDüzenle

apiVersion: tekton.dev/v1beta1

kind: Pipeline

metadata:

name: ml-pipeline

namespace: rhods-project

spec:

params:

- name: storage

default: "10Gi"

base/rbac.yaml

yaml

KopyalaDüzenle

apiVersion: rbac.authorization.k8s.io/v1

kind: RoleBinding

metadata:

name: rhods-project-admin

namespace: rhods-project

subjects:

- kind: Group

name: data-scientists

apiGroup: rbac.authorization.k8s.io

roleRef:

kind: ClusterRole

name: admin

apiGroup: rbac.authorization.k8s.io

base/kustomization.yaml

yaml

KopyalaDüzenle

apiVersion: kustomize.config.k8s.io/v1beta1

kind: Kustomization

resources:

- namespace.yaml

- workbenches/team-notebook-1.yaml

- workbenches/team-notebook-2.yaml

- model-serving/fraud-detector.yaml

- model-serving/image-classifier.yaml

- pipelines/ml-pipeline.yaml

- rbac.yaml

commonLabels:

app.kubernetes.io/managed-by: kustomize

2. overlays/data-scientists

overlays/data-scientists/kustomization.yaml

yaml

KopyalaDüzenle

apiVersion: kustomize.config.k8s.io/v1beta1

kind: Kustomization

bases:

- ../../base

patchesStrategicMerge:

- patch-notebook.yaml
- patch-runtime.yaml

commonLabels:

rhods-group: data-scientists

overlays/data-scientists/patch-notebook.yaml

yaml

KopyalaDüzenle

apiVersion: kubeflow.org/v1

kind: Notebook

metadata:

name: team-notebook-1

spec:

template:

spec:

containers:

- name: team-notebook-1

resources:

limits:

cpu: "4"

memory: "16Gi"

overlays/data-scientists/patch-runtime.yaml

yaml

KopyalaDüzenle

apiVersion: serving.kserve.io/v1beta1

kind: InferenceService

metadata:

name: fraud-detector

spec:

predictor:

model:

resources:

limits:

cpu: "2"

memory: "8Gi"

3. overlays/ml-engineers

overlays/ml-engineers/kustomization.yaml

yaml

KopyalaDüzenle

apiVersion: kustomize.config.k8s.io/v1beta1

kind: Kustomization

bases:

- ../../base

patchesStrategicMerge:

- patch-notebook.yaml

commonLabels:

rhods-group: ml-engineers

overlays/ml-engineers/patch-notebook.yaml

yaml

KopyalaDüzenle

apiVersion: kubeflow.org/v1

kind: Notebook

metadata:

name: team-notebook-2

spec:

template:

spec:

containers:

- name: team-notebook-2

resources:

limits:

cpu: "6"

memory: "24Gi"

Sırada: 📄 **Tüm bu dosya ve izinleri otomatik oluşturan bir bash script.** Onu da hemen sunuyorum.
Devam edelim mi?

Şunu dedin:

evet

ChatGPT:

İşte yukarıdaki tüm base/ ve overlays/ yapısını otomatik olarak oluşturan bir **bash script**:

📄 **setup-kustomize-structure.sh**

bash

KopyalaDüzenle

#!/bin/bash

set -e

```
ROOT_DIR="rhods-gitops-kustomize"
mkdir -p $ROOT_DIR/base/{workbenches,model-serving,pipelines}
mkdir -p $ROOT_DIR/overlays/data-scientists
mkdir -p $ROOT_DIR/overlays/ml-engineers
```

```
# === BASE ===
```

```
cat > $ROOT_DIR/base/namespace.yaml <<EOF
```

```
apiVersion: v1
```

```
kind: Namespace
```

```
metadata:
```

```
  name: rhods-project
```

```
  labels:
```

```
    opendatahub.io/dashboard: "true"
```

```
EOF
```

```
cat > $ROOT_DIR/base/workbenches/team-notebook-1.yaml <<EOF
```

```
apiVersion: kubeflow.org/v1
```

```
kind: Notebook
```

```
metadata:
```

```
  name: team-notebook-1
```

```
  namespace: rhods-project
```

```
spec:
```

```
  template:
```

```
    spec:
```

```
      containers:
```

```
        - name: team-notebook-1
```

```
          image: s2i-minimal-notebook
```

```
          resources:
```

```
            limits:
```



```
    cpu: "2"
    memory: "8Gi"
EOF
```

```
cat > $ROOT_DIR/base/workbenches/team-notebook-2.yaml <<EOF
apiVersion: kubeflow.org/v1
kind: Notebook
metadata:
  name: team-notebook-2
  namespace: rhods-project
spec:
  template:
    spec:
      containers:
      - name: team-notebook-2
        image: pytorch-gpu
        resources:
          limits:
            cpu: "2"
            memory: "8Gi"
EOF
```

```
cat > $ROOT_DIR/base/model-serving/fraud-detector.yaml <<EOF
apiVersion: serving.kserve.io/v1beta1
kind: InferenceService
metadata:
  name: fraud-detector
  namespace: rhods-project
spec:
  predictor:
    model:
```

```
    modelFormat:
      name: onnx
    runtime: ovms-runtime
    storageUri: s3://models/fraud-detector
EOF
```

```
cat > $ROOT_DIR/base/model-serving/image-classifier.yaml <<EOF
apiVersion: serving.kserve.io/v1beta1
kind: InferenceService
metadata:
  name: image-classifier
  namespace: rhods-project
spec:
  predictor:
    model:
      modelFormat:
        name: tensorrt
      runtime: triton-runtime
      storageUri: s3://models/image-classifier
EOF
```

```
cat > $ROOT_DIR/base/pipelines/ml-pipeline.yaml <<EOF
apiVersion: tekton.dev/v1beta1
kind: Pipeline
metadata:
  name: ml-pipeline
  namespace: rhods-project
spec:
  params:
    - name: storage
      default: "10Gi"
```

EOF

```
cat > $ROOT_DIR/base/rbac.yaml <<EOF
```

```
apiVersion: rbac.authorization.k8s.io/v1
```

```
kind: RoleBinding
```

```
metadata:
```

```
  name: rhods-project-admin
```

```
  namespace: rhods-project
```

```
subjects:
```

```
- kind: Group
```

```
  name: data-scientists
```

```
  apiGroup: rbac.authorization.k8s.io
```

```
roleRef:
```

```
  kind: ClusterRole
```

```
  name: admin
```

```
  apiGroup: rbac.authorization.k8s.io
```

EOF

```
cat > $ROOT_DIR/base/kustomization.yaml <<EOF
```

```
apiVersion: kustomize.config.k8s.io/v1beta1
```

```
kind: Kustomization
```

```
resources:
```

```
- namespace.yaml
```

```
- workbenches/team-notebook-1.yaml
```

```
- workbenches/team-notebook-2.yaml
```

```
- model-serving/fraud-detector.yaml
```

```
- model-serving/image-classifier.yaml
```

```
- pipelines/ml-pipeline.yaml
```

```
- rbac.yaml
```

```
commonLabels:
  app.kubernetes.io/managed-by: kustomize
EOF
```

```
# === OVERLAYS: data-scientists ===
```

```
cat > $ROOT_DIR/overlays/data-scientists/kustomization.yaml <<EOF
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization
```

```
bases:
- ../../base
```

```
patchesStrategicMerge:
- patch-notebook.yaml
- patch-runtime.yaml
```

```
commonLabels:
  rhods-group: data-scientists
EOF
```

```
cat > $ROOT_DIR/overlays/data-scientists/patch-notebook.yaml <<EOF
apiVersion: kubeflow.org/v1
kind: Notebook
metadata:
  name: team-notebook-1
spec:
  template:
    spec:
      containers:
        - name: team-notebook-1
```

```
resources:
  limits:
    cpu: "4"
    memory: "16Gi"
```

EOF

```
cat > $ROOT_DIR/overlays/data-scientists/patch-runtime.yaml <<EOF
```

```
apiVersion: serving.kserve.io/v1beta1
```

```
kind: InferenceService
```

```
metadata:
```

```
  name: fraud-detector
```

```
spec:
```

```
  predictor:
```

```
    model:
```

```
      resources:
```

```
        limits:
```

```
          cpu: "2"
```

```
          memory: "8Gi"
```

EOF

```
# === OVERLAYS: ml-engineers ===
```

```
cat > $ROOT_DIR/overlays/ml-engineers/kustomization.yaml <<EOF
```

```
apiVersion: kustomize.config.k8s.io/v1beta1
```

```
kind: Kustomization
```

```
bases:
```

```
- ../../base
```

```
patchesStrategicMerge:
```

```
- patch-notebook.yaml
```

commonLabels:

rhods-group: ml-engineers

EOF

```
cat > $ROOT_DIR/overlays/ml-engineers/patch-notebook.yaml <<EOF
```

apiVersion: kubeflow.org/v1

kind: Notebook

metadata:

name: team-notebook-2

spec:

template:

spec:

containers:

- name: team-notebook-2

resources:

limits:

cpu: "6"

memory: "24Gi"

EOF

```
echo "✅ Yapı '$(ROOT_DIR)/' dizini altında oluşturuldu."
```



Kullanım

1. Script'i bir dosyaya kaydet:

bash

KopyalaDüzenle

nano setup-kustomize-structure.sh

2. İçeriği yapıştır ve kaydet.
3. Çalıştırılabilir yap:

bash

KopyalaDüzenle

```
chmod +x setup-kustomize-structure.sh
```

4. Çalıştır:

```
bash
```

KopyalaDüzenle

```
./setup-kustomize-structure.sh
```

İsteğe göre:

- ApplicationSet dosyası da ekleyebilirim.
- Her overlay için ArgoCD Application örneği çıkartabilirim.

Bir sonraki adım ne olsun?

Şunu dedin:

yeni bir kullanıcı veya grup için overlay altında örneğin/overlay/newuser/ ve altında bir json request file ile user workbench ve model gibi istek bilgilerine göre kullanıcı dosyaları otomatik oluşsa

ChatGPT:

Evet, bu ihtiyacını karşılamak için aşağıdaki çözümü öneriyorum:

Hedef

Bir JSON request dosyası verildiğinde, aşağıdaki işlemler otomatik yapılacaktır:

- overlays/<kullanıcı-adı>/ dizini oluşturulacaktır
- Bu dizin altına:
 - kustomization.yaml
 - İlgili patch-notebook.yaml, patch-runtime.yaml, patch-pipeline.yaml dosyaları
 - Orijinal request.json saklanacaktır
- Kullanıcının:
 - Workbench
 - Model (InferenceService)
 - Pipeline
 - RBAC tanımlamaları base/ şablonlarını *patch* ederek yapılacaktır.

 **Örnek JSON İstek (örnek path: overlays/newuser/request.json)**

json

KopyalaDüzenle

```
{
  "type": "user",
  "name": "newuser",
  "resources": {
    "workbenches": [
      {
        "name": "newuser-notebook",
        "image": "minimal-py39",
        "cpu": "1",
        "memory": "2Gi"
      }
    ],
    "models": [
      {
        "name": "newuser-model",
        "format": "onnx",
        "runtime": "ovms",
        "cpu": "1",
        "memory": "2Gi"
      }
    ]
  }
}
```

📄 Script Önerisi: generate-overlay.sh

Bu script:

- overlays/<name>/ klasörü oluşturur.
- kustomization.yaml oluşturur.
- JSON'dan aldığı değerlerle:

- patch-notebook.yaml
- patch-runtime.yaml
- Gerekirse patch-rbac.yaml ve patch-pipeline.yaml ekler.

Kullanım Senaryosu

bash

KopyalaDüzenle

./generate-overlay.sh overlays/newuser/request.json

Bu komut:

- overlays/newuser/ klasörünü yaratır.
- request.json'daki bilgilere göre patch dosyalarını üretir.
- kustomization.yaml içinde doğru referansları tanımlar.

Devam Edelim mi?

İstersen bu scripti şimdi senin için oluşturayım ve içinde örnek bir request.json ile birlikte çalışır hale getireyim.

Script çıktısı da doğrudan kullanılabilir olacak.