# Release Engineering for ML Applications

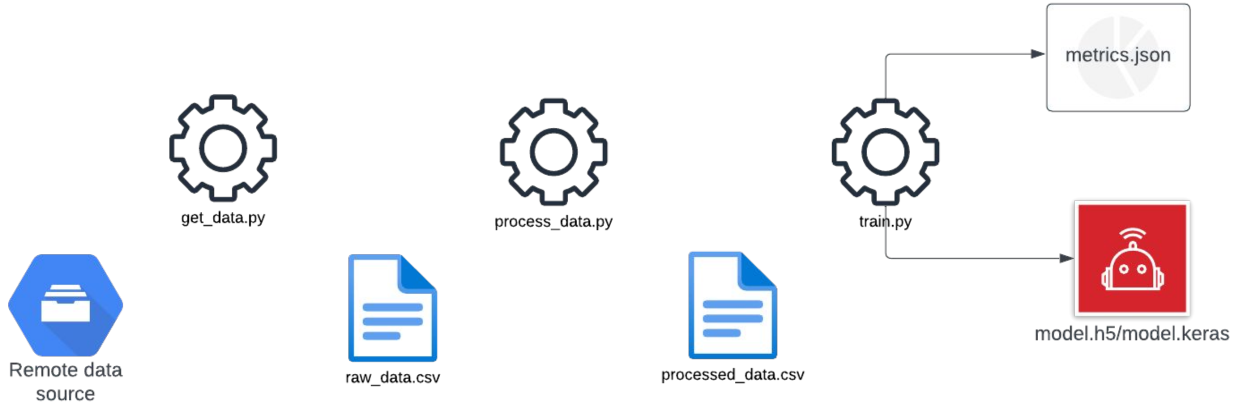
Team 14: Dani, Justin, Nadine, Yang Li

28 June 2024

## Introduction

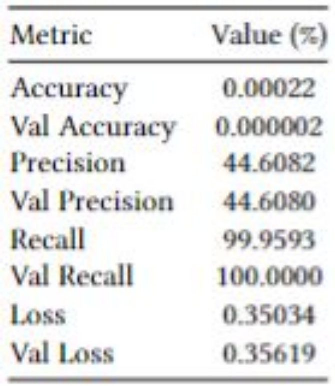
## 1. ML Pipeline

- DVC
- AWS S3
- Poetry package/dependency manager



## 1. ML Pipeline: artifacts

- Trained model
- Raw and processed data
- metrics



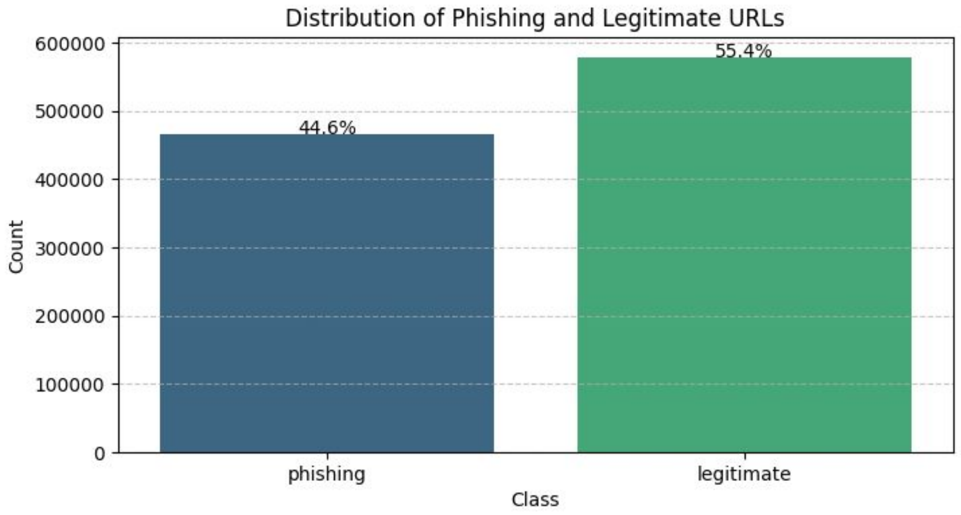
| Metric | Value (%) |
|---|---|
| Accuracy | 0.00022 |
| Val Accuracy | 0.000002 |
| Precision | 44.6082 |
| Val Precision | 44.6080 |
| Recall | 99.9593 |
| Val Recall | 100.0000 |
| Loss | 0.35034 |
| Val Loss | 0.35619 |

## 1. ML Pipeline: linters

- Pylint & Flake8
- Allow common naming conventions
- Discourage ‘bad’ naming
- Handle more exceptions
- Display only warnings with high confidence levels

## 2. ML Testing Design: features and data

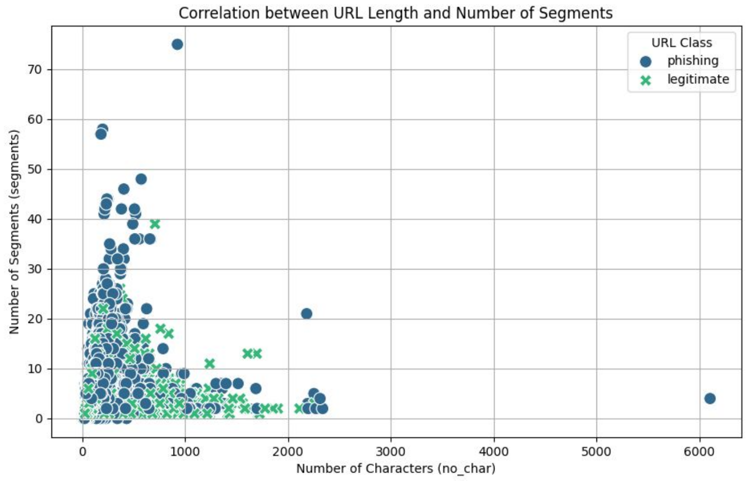
- View distribution of the data



TU Delft

## 2. ML Testing Design: Features and Data

- View distribution of the data
- Engineered features

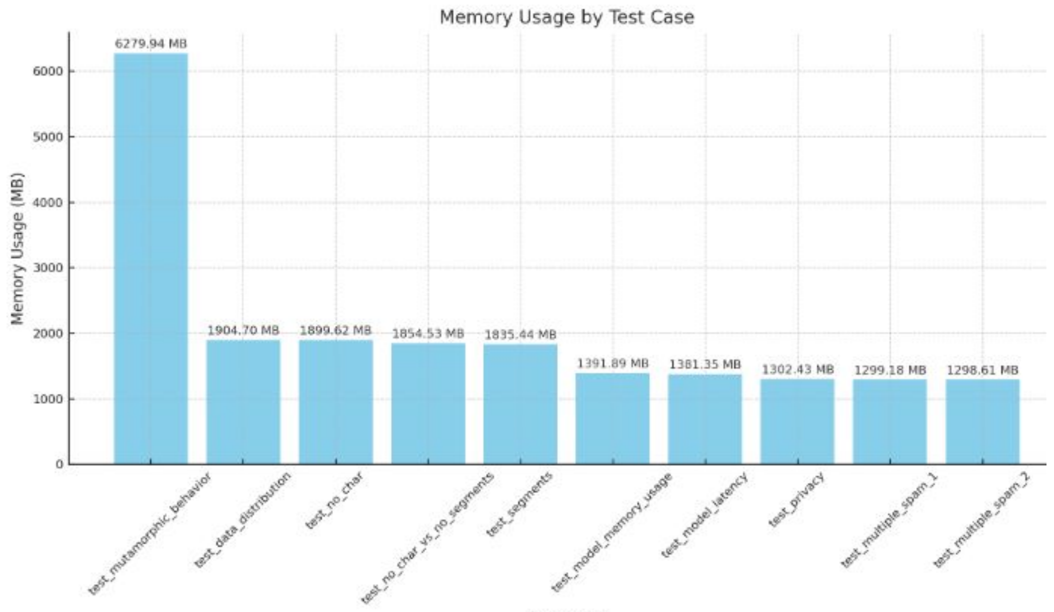


## 2. ML Testing Design: Mutamorphic tests

- Check for non-deterministic behaviour

## 2. ML Testing Design: Monitoring Tests

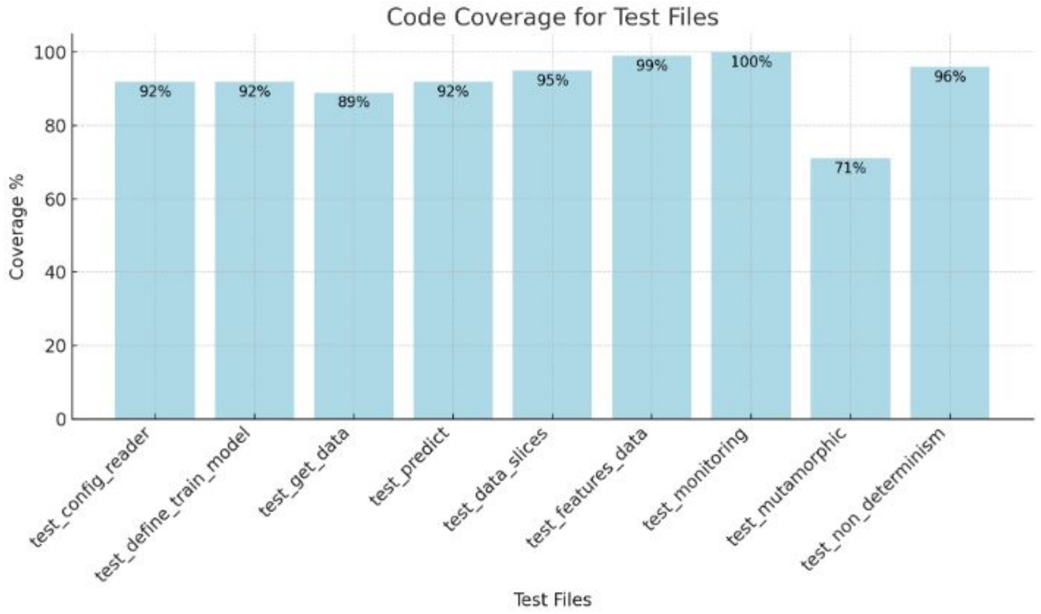
- Ram usage per method



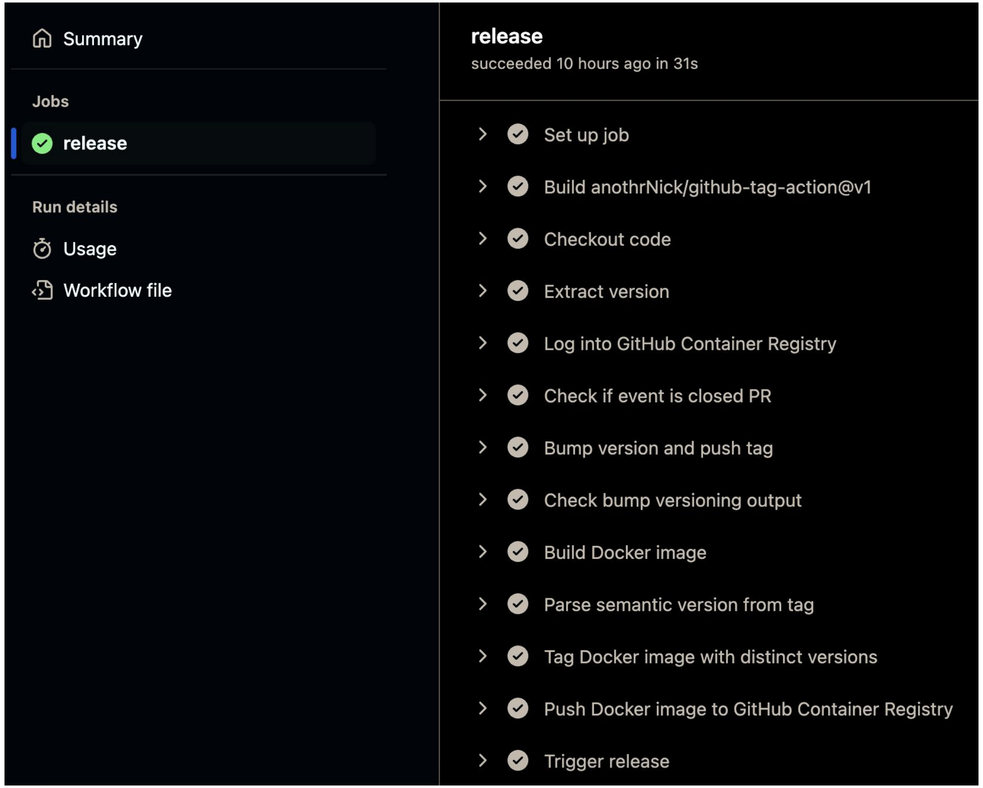
TUDelft

## 2. ML Testing Design: Test Adequacy

- Statements, covered, missed



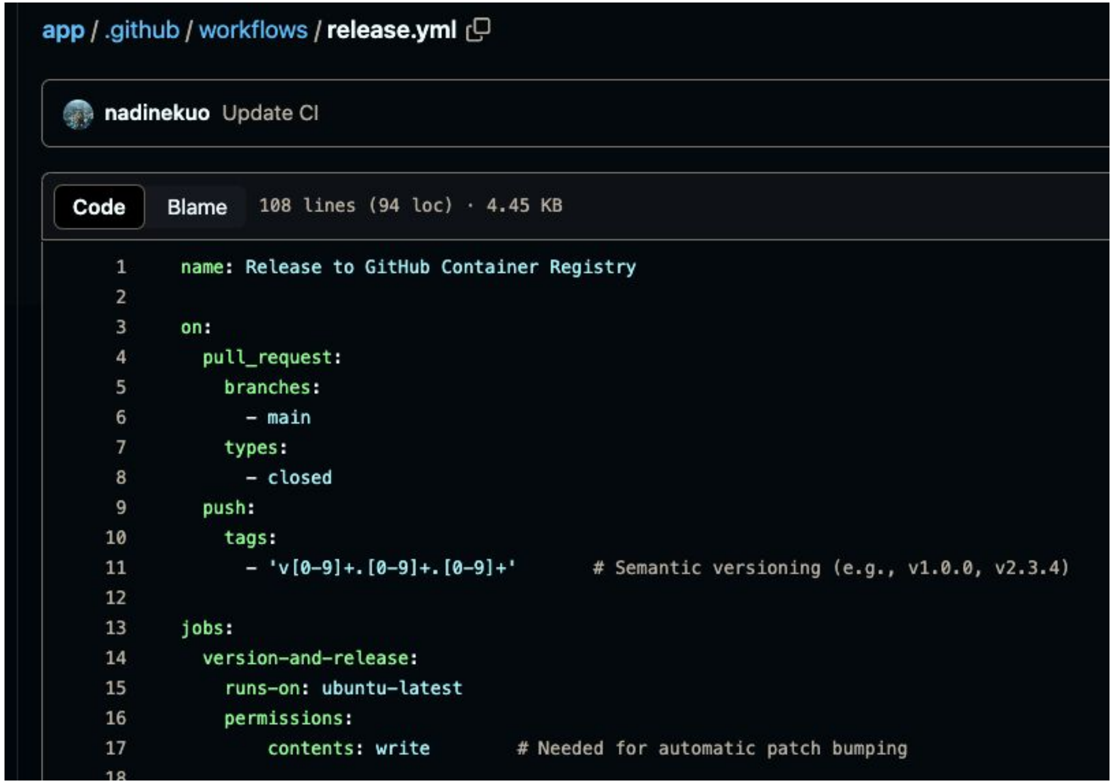
# 3. Release Pipeline: Container Image (app)



release
succeeded 10 hours ago in 31s

- Set up job
- Build anothrNick/github-tag-action@v1
- Checkout code
- Extract version
- Log into GitHub Container Registry
- Check if event is closed PR
- Bump version and push tag
- Check bump versioning output
- Build Docker image
- Parse semantic version from tag
- Tag Docker image with distinct versions
- Push Docker image to GitHub Container Registry
- Trigger release

## 3. Release Pipeline: Container Image (app)


app / .github / workflows / release.yml

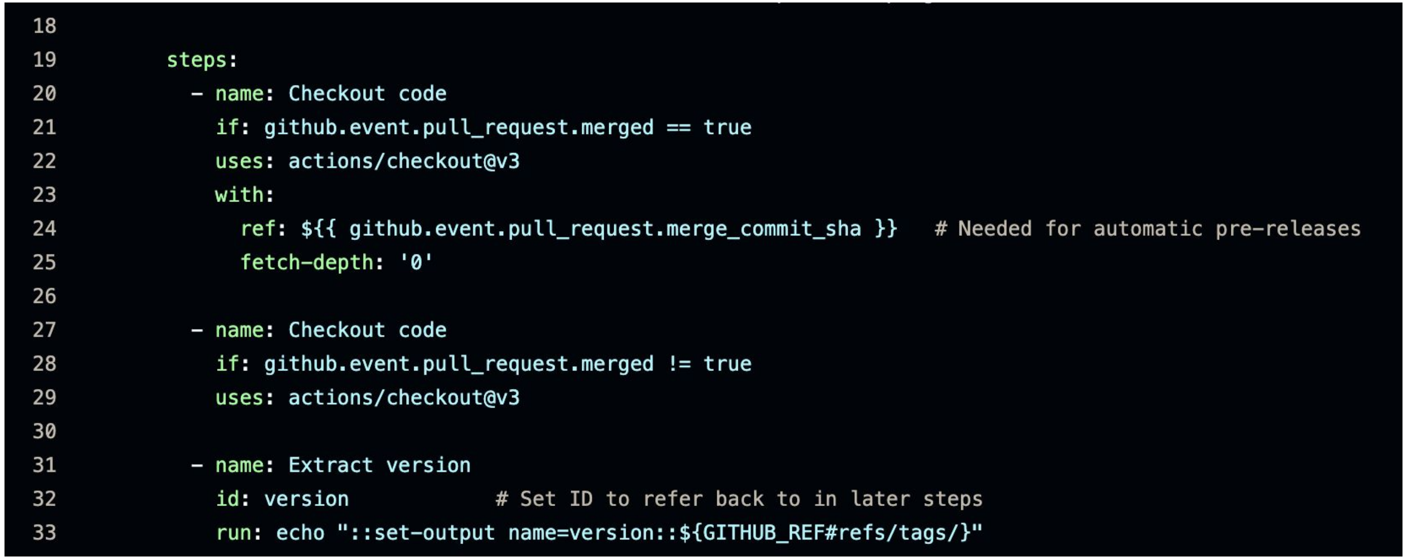
nadinekuo Update CI

Code Blame 108 lines (94 loc) · 4.45 KB

```yaml
name: Release to GitHub Container Registry

on:
  pull_request:
    branches:
      - main
    types:
      - closed
  push:
    tags:
      - 'v[0-9]+.[0-9]+.[0-9]+'       # Semantic versioning (e.g., v1.0.0, v2.3.4)

jobs:
  version-and-release:
    runs-on: ubuntu-latest
    permissions:
        contents: write        # Needed for automatic patch bumping
```

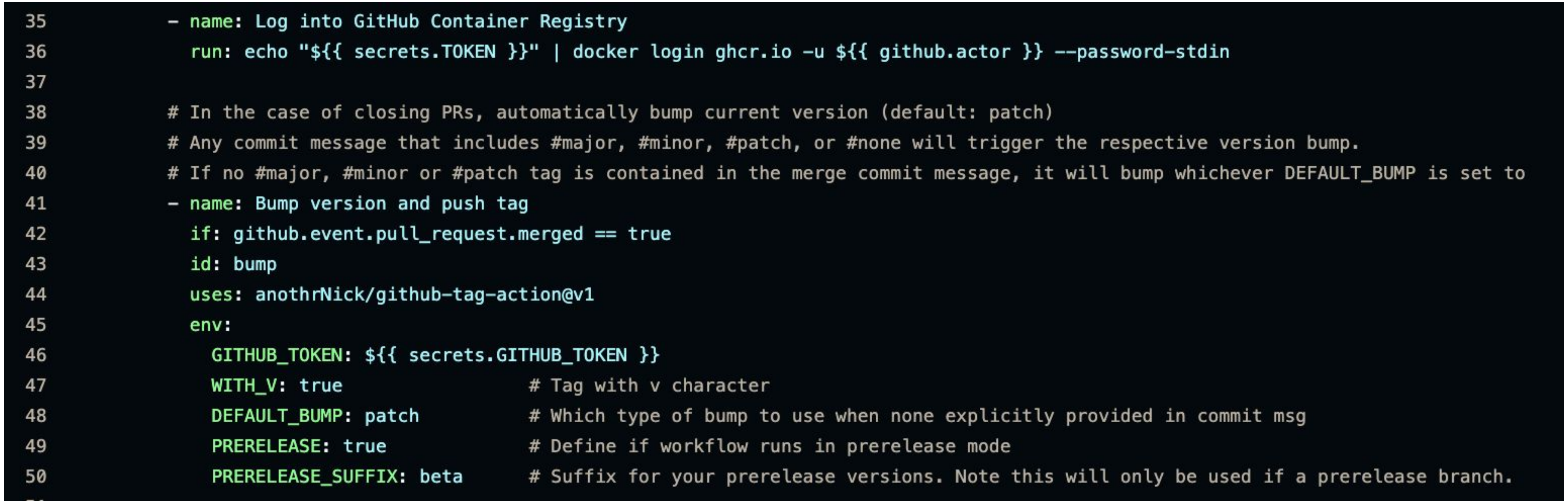
TU Delft

# 3. Release Pipeline: Container Image (app)



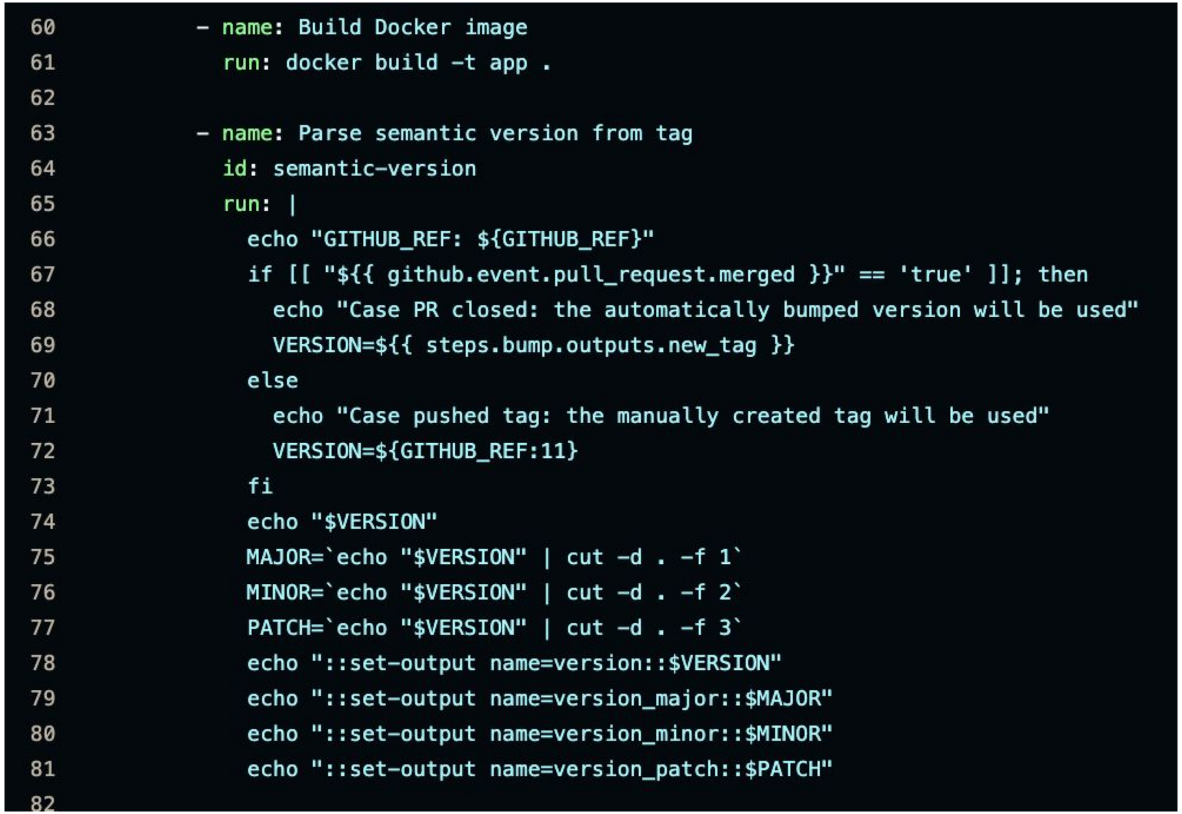
```yaml

    steps:
      - name: Checkout code
        if: github.event.pull_request.merged == true
        uses: actions/checkout@v3
        with:
          ref: ${{ github.event.pull_request.merge_commit_sha }}   # Needed for automatic pre-releases
          fetch-depth: '0'

      - name: Checkout code
        if: github.event.pull_request.merged != true
        uses: actions/checkout@v3

      - name: Extract version
        id: version            # Set ID to refer back to in later steps
        run: echo "::set-output name=version::${GITHUB_REF#refs/tags/}"
```

# 3. Release Pipeline: Container Image (app)



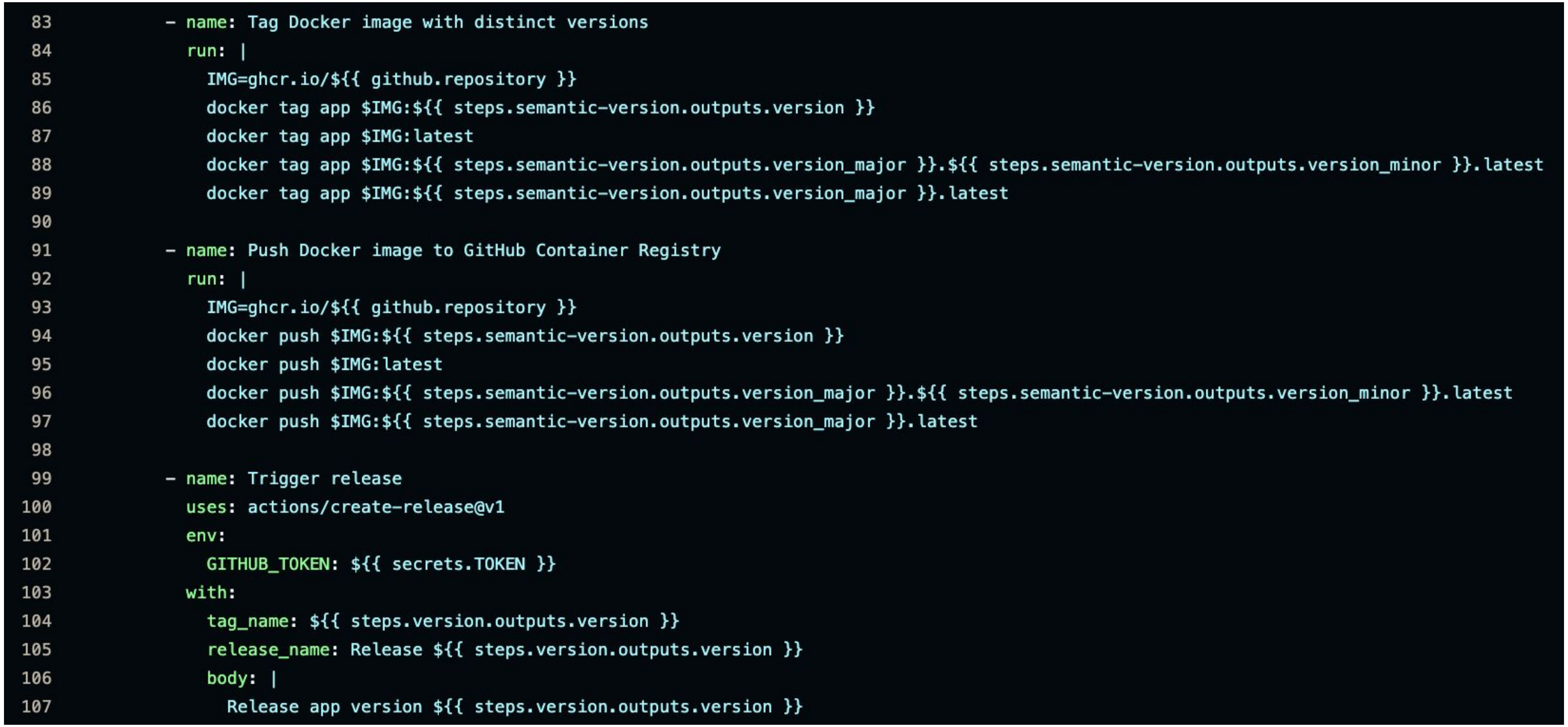
```
      - name: Log into GitHub Container Registry
        run: echo "${{ secrets.TOKEN }}" | docker login ghcr.io -u ${{ github.actor }} --password-stdin

      # In the case of closing PRs, automatically bump current version (default: patch)
      # Any commit message that includes #major, #minor, #patch, or #none will trigger the respective version bump.
      # If no #major, #minor or #patch tag is contained in the merge commit message, it will bump whichever DEFAULT_BUMP is set to
      - name: Bump version and push tag
        if: github.event.pull_request.merged == true
        id: bump
        uses: anothrNick/github-tag-action@v1
        env:
          GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
          WITH_V: true                # Tag with v character
          DEFAULT_BUMP: patch         # Which type of bump to use when none explicitly provided in commit msg
          PRERELEASE: true            # Define if workflow runs in prerelease mode
          PRERELEASE_SUFFIX: beta     # Suffix for your prerelease versions. Note this will only be used if a prerelease branch.
```

# 3. Release Pipeline: Container Image (app)



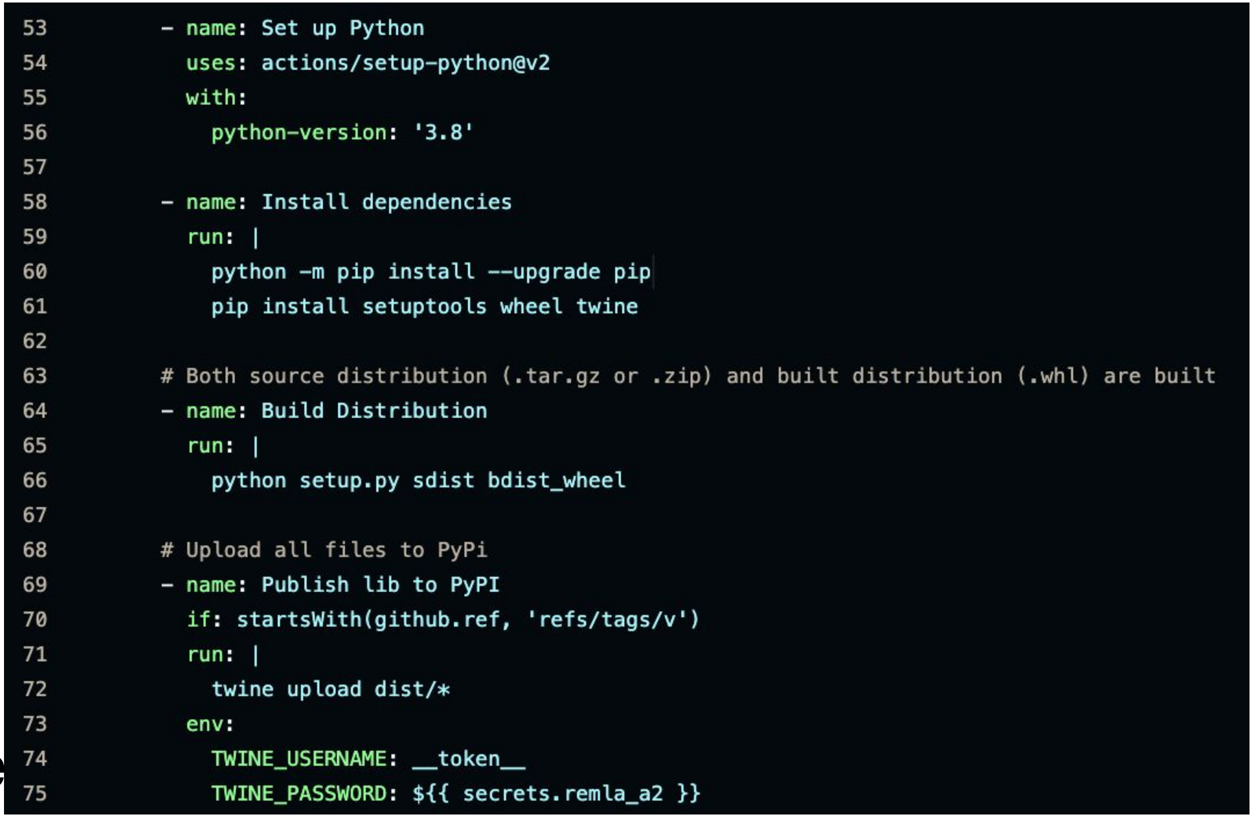
```
      - name: Build Docker image
        run: docker build -t app .

      - name: Parse semantic version from tag
        id: semantic-version
        run: |
          echo "GITHUB_REF: ${GITHUB_REF}"
          if [[ "${{ github.event.pull_request.merged }}" == 'true' ]]; then
            echo "Case PR closed: the automatically bumped version will be used"
            VERSION=${{ steps.bump.outputs.new_tag }}
          else
            echo "Case pushed tag: the manually created tag will be used"
            VERSION=${GITHUB_REF:11}
          fi
          echo "$VERSION"
          MAJOR=`echo "$VERSION" | cut -d . -f 1`
          MINOR=`echo "$VERSION" | cut -d . -f 2`
          PATCH=`echo "$VERSION" | cut -d . -f 3`
          echo "::set-output name=version::$VERSION"
          echo "::set-output name=version_major::$MAJOR"
          echo "::set-output name=version_minor::$MINOR"
          echo "::set-output name=version_patch::$PATCH"

```

## 3. Release Pipeline: Container Image (app)



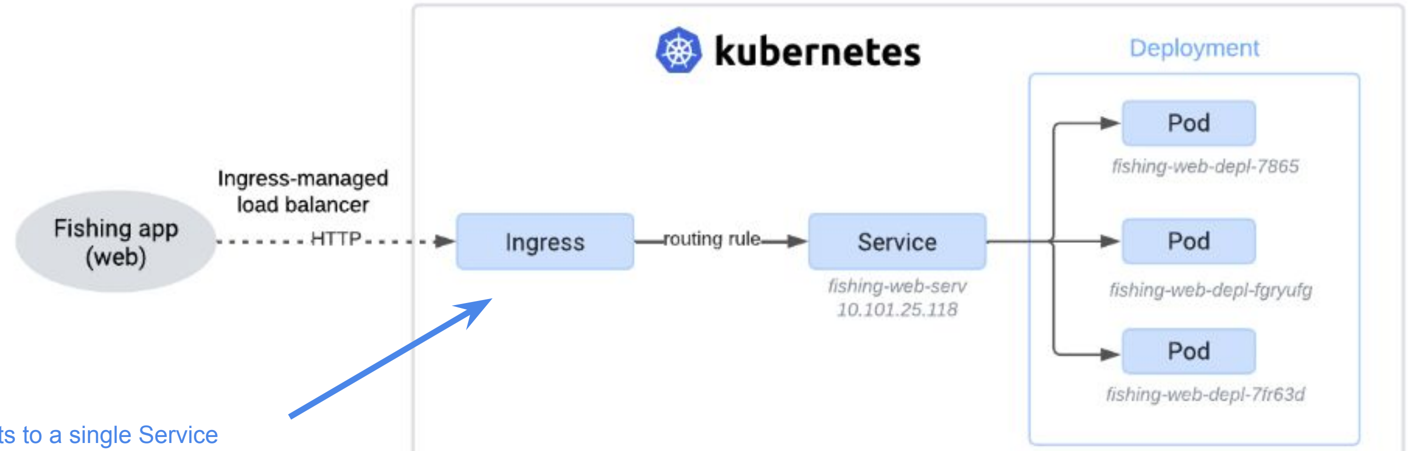
```yaml
      - name: Tag Docker image with distinct versions
        run: |
          IMG=ghcr.io/${{ github.repository }}
          docker tag app $IMG:${{ steps.semantic-version.outputs.version }}
          docker tag app $IMG:latest
          docker tag app $IMG:${{ steps.semantic-version.outputs.version_major }}.${{ steps.semantic-version.outputs.version_minor }}.latest
          docker tag app $IMG:${{ steps.semantic-version.outputs.version_major }}.latest

      - name: Push Docker image to GitHub Container Registry
        run: |
          IMG=ghcr.io/${{ github.repository }}
          docker push $IMG:${{ steps.semantic-version.outputs.version }}
          docker push $IMG:latest
          docker push $IMG:${{ steps.semantic-version.outputs.version_major }}.${{ steps.semantic-version.outputs.version_minor }}.latest
          docker push $IMG:${{ steps.semantic-version.outputs.version_major }}.latest

      - name: Trigger release
        uses: actions/create-release@v1
        env:
          GITHUB_TOKEN: ${{ secrets.TOKEN }}
        with:
          tag_name: ${{ steps.version.outputs.version }}
          release_name: Release ${{ steps.version.outputs.version }}
          body: |
            Release app version ${{ steps.version.outputs.version }}
```

## 3. Release Pipeline: Software Package (lib-ml)

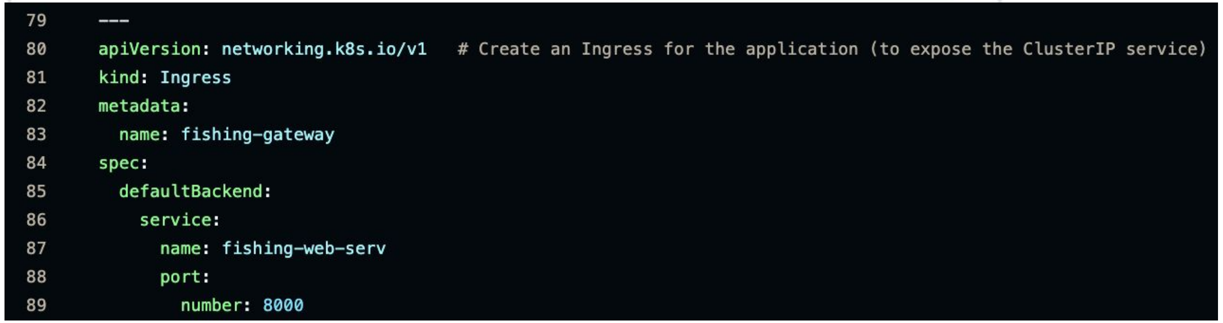


```yaml
      - name: Set up Python
        uses: actions/setup-python@v2
        with:
          python-version: '3.8'

      - name: Install dependencies
        run: |
          python -m pip install --upgrade pip
          pip install setuptools wheel twine

      # Both source distribution (.tar.gz or .zip) and built distribution (.whl) are built
      - name: Build Distribution
        run: |
          python setup.py sdist bdist_wheel

      # Upload all files to PyPi
      - name: Publish lib to PyPI
        if: startsWith(github.ref, 'refs/tags/v')
        run: |
          twine upload dist/*
        env:
          TWINE_USERNAME: __token__
          TWINE_PASSWORD: ${{ secrets.remla_a2 }}
```

## 4. K8s Deployment



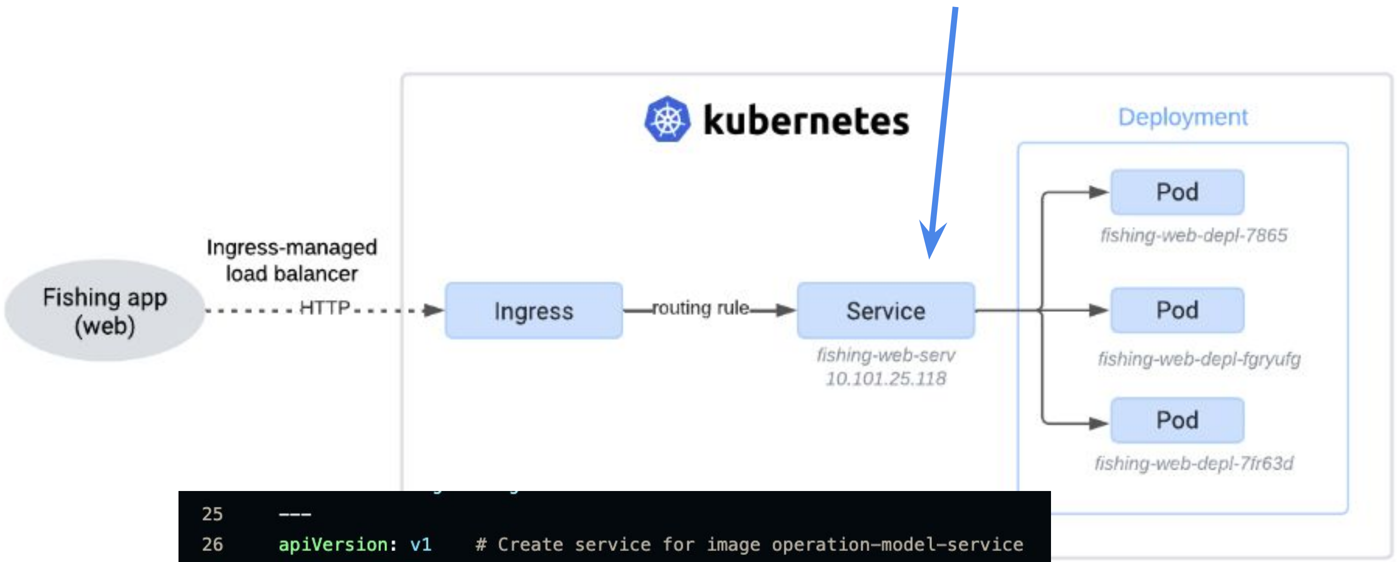
- Routes requests to a single Service
- Ingress Controller determines relevant Service port

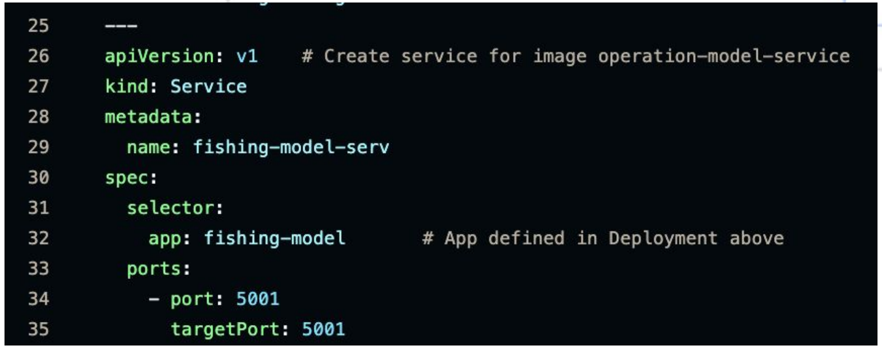


```yaml
---
apiVersion: networking.k8s.io/v1   # Create an Ingress for the application (to expose the ClusterIP service)
kind: Ingress
metadata:
  name: fishing-gateway
spec:
  defaultBackend:
    service:
      name: fishing-web-serv
      port:
        number: 8000
```

## 4. K8s Deployment

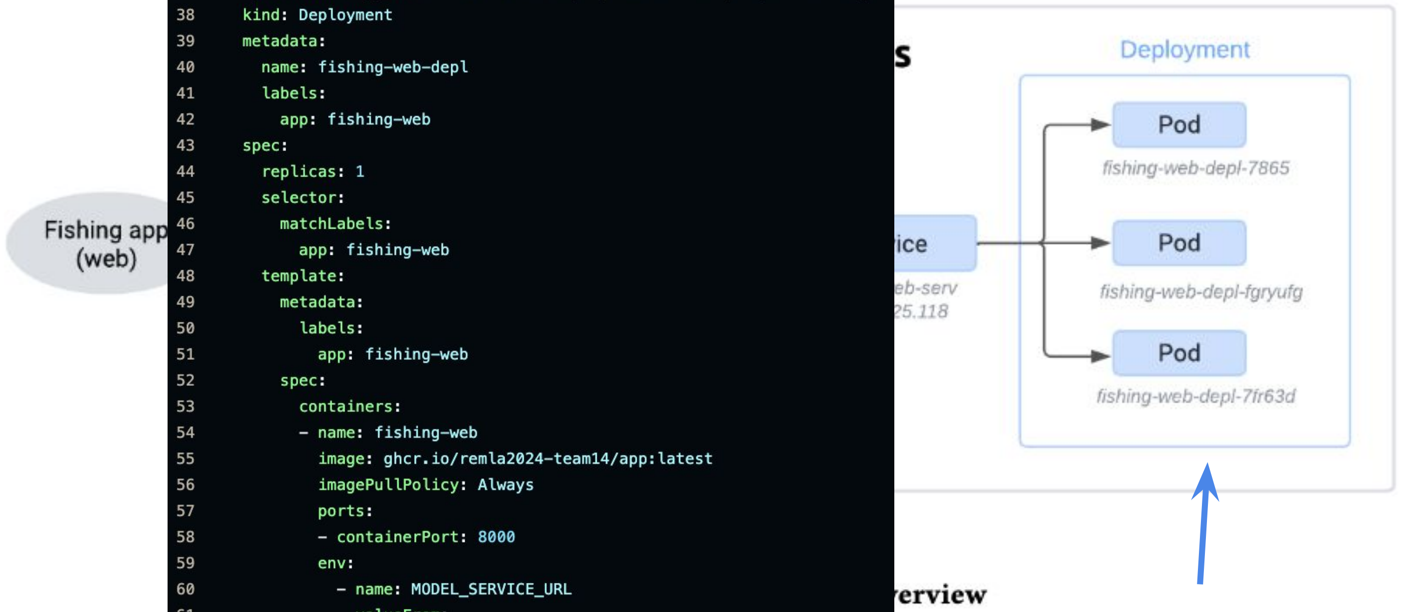
- Prevent internal Pods from being accessed publicly
- Listen on specified port (fixed)
- Forward requests to containers in Pods

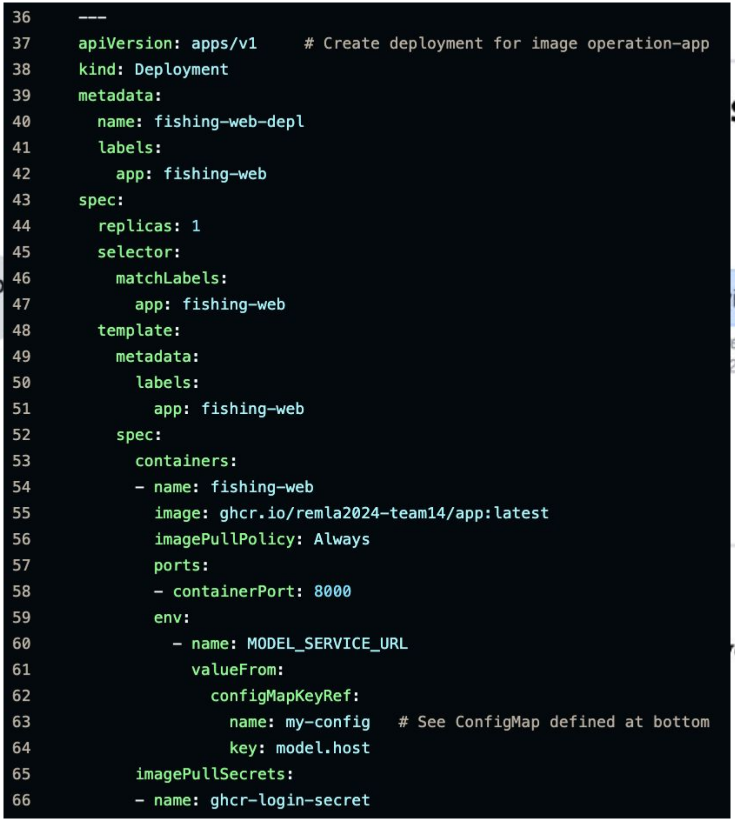




```yaml
---
apiVersion: v1    # Create service for image operation-model-service
kind: Service
metadata:
  name: fishing-model-serv
spec:
  selector:
    app: fishing-model       # App defined in Deployment above
  ports:
    - port: 5001
      targetPort: 5001
```

# 4. K8s Deployment



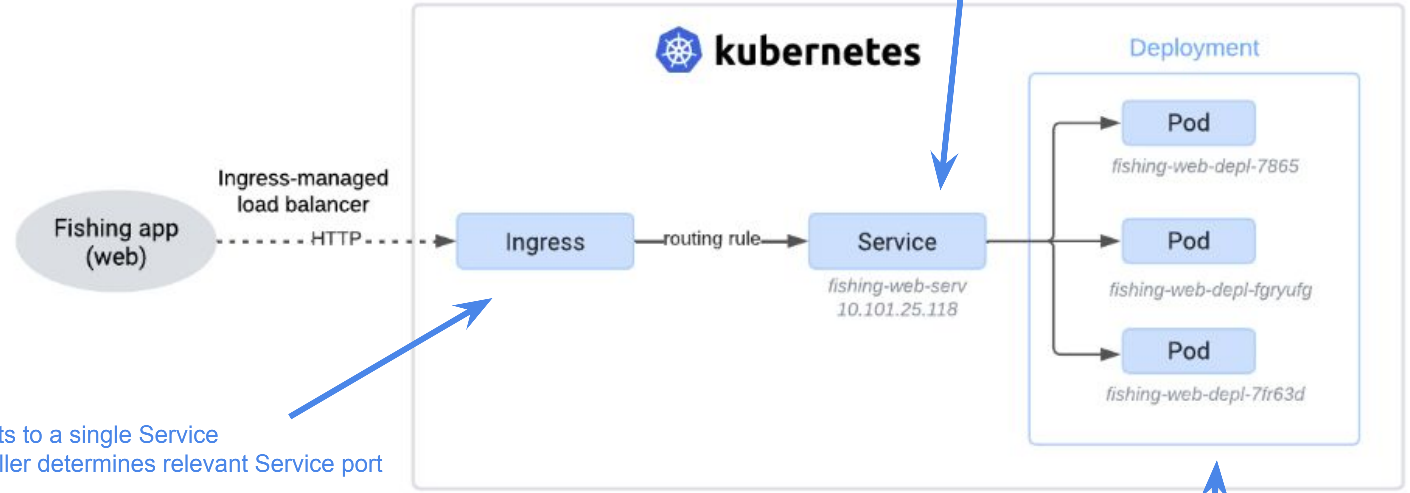


```yaml
---
apiVersion: apps/v1     # Create deployment for image operation-app
kind: Deployment
metadata:
  name: fishing-web-depl
  labels:
    app: fishing-web
spec:
  replicas: 1
  selector:
    matchLabels:
      app: fishing-web
  template:
    metadata:
      labels:
        app: fishing-web
    spec:
      containers:
      - name: fishing-web
        image: ghcr.io/remla2024-team14/app:latest
        imagePullPolicy: Always
        ports:
        - containerPort: 8000
        env:
          - name: MODEL_SERVICE_URL
            valueFrom:
              configMapKeyRef:
                name: my-config   # See ConfigMap defined at bottom
                key: model.host
      imagePullSecrets:
      - name: ghcr-login-secret
```

- Specification of desired state of Pods
- K8s ensures this state is retained + allows for scaling up or down replicas of Pods

# 4. K8s Deployment

- Prevent internal Pods from being accessed publicly
- Listen on specified port (fixed)
- Forward requests to containers in Pods



- Routes requests to a single Service
- Ingress Controller determines relevant Service port

**Figure 2: Kubernetes Deployment Overview**

- Specification of desired state of Pods
- K8s ensures this state is retained + allows for scaling up or down replicas of Pods

TU Delft

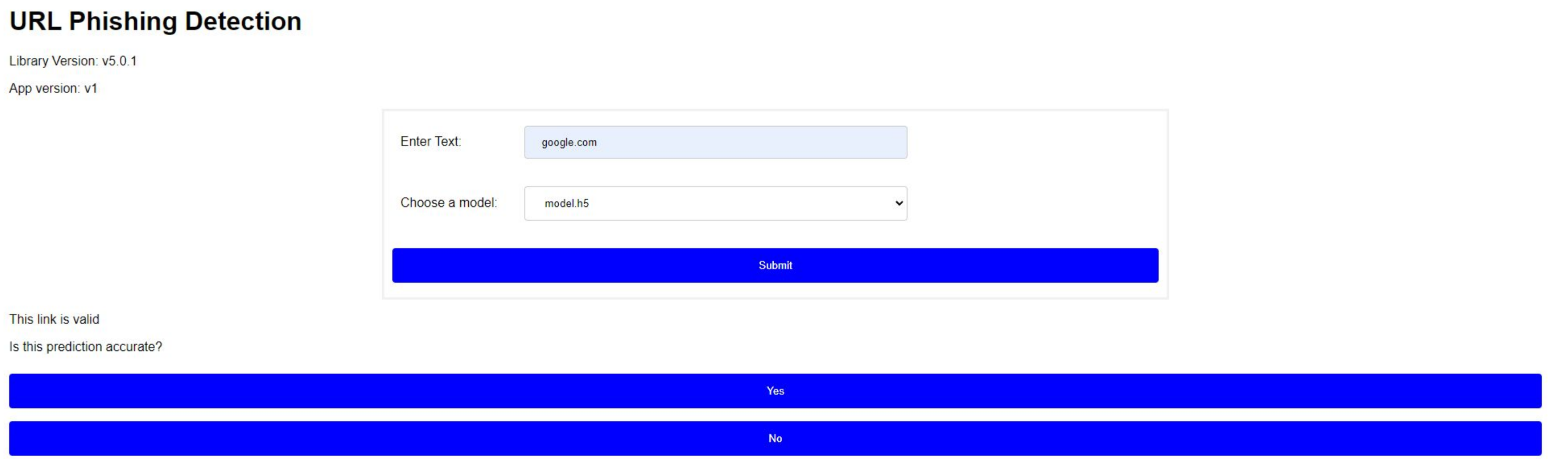
## 5. Experimental Setup

Requirements:

- Istio Service Mesh that deploys two versions of app-frontend and app-service.
  - Prometheus annotations to scrape.
- Verifiable hypothesis with a reject/validate criterion.
- Prometheus scrapes the relevant metrics.
  - Summary, counter, gauge, histogram.
  - Metrics about inference time, user interaction metrics, etc…
- Grafana displays results
- AlertManager sends an email for extreme cases (example: too many calls)

Hypothesis: *“Given a version of the web-app that returns binary predictions (valid/phishing) to the user and a version that returns probabilities instead, the user will prefer the version of the web-app that returns binary predictions”*

TU Delft


URL Phishing Detection

Library Version: v5.0.1

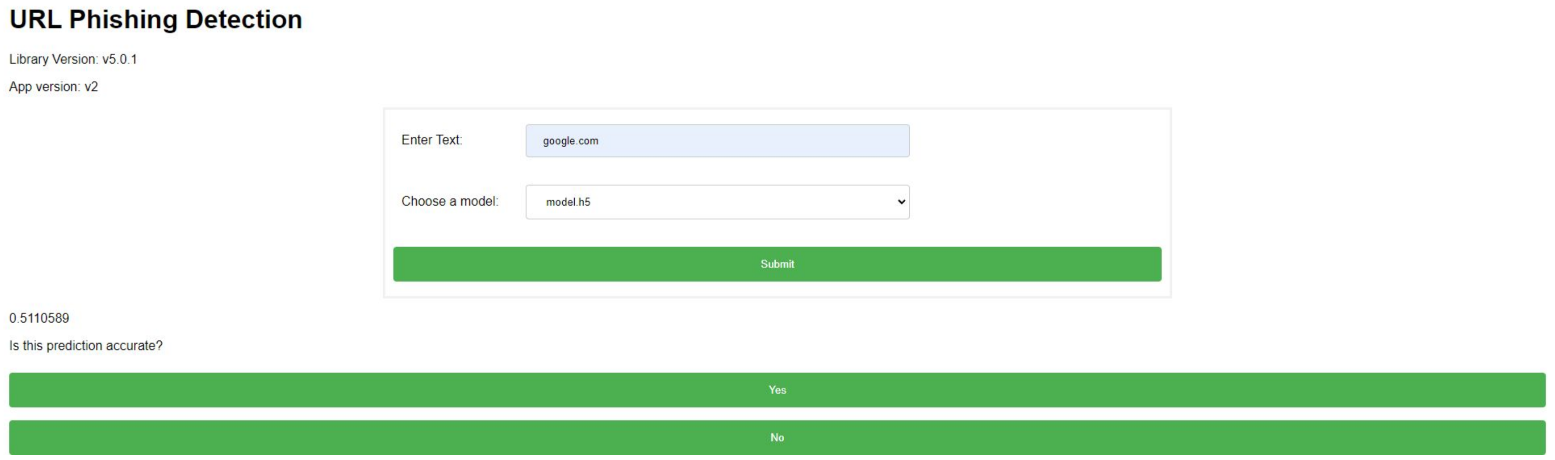
App version: v1

Enter Text: google.com

Choose a model: model.h5

This link is valid

Is this prediction accurate?


URL Phishing Detection

Library Version: v5.0.1

App version: v2

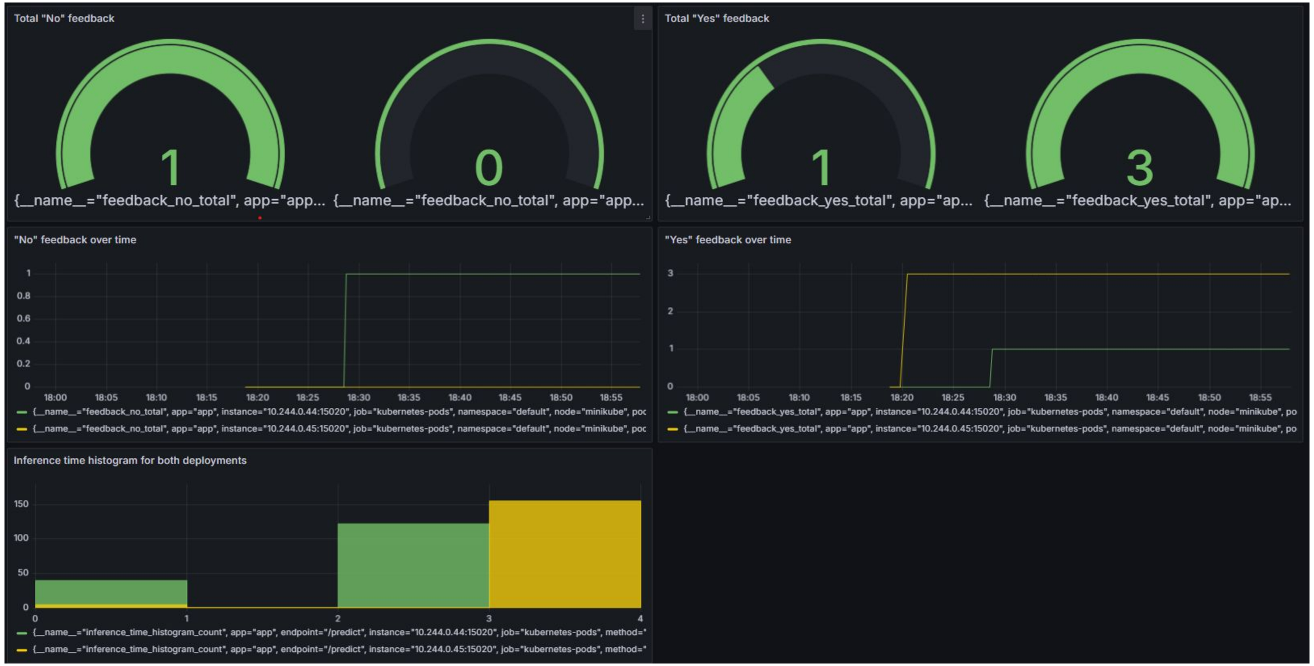
Enter Text: google.com

Choose a model: model.h5

0.5110589

Is this prediction accurate?

# 5. Experimental Setup: Prometheus and Grafana



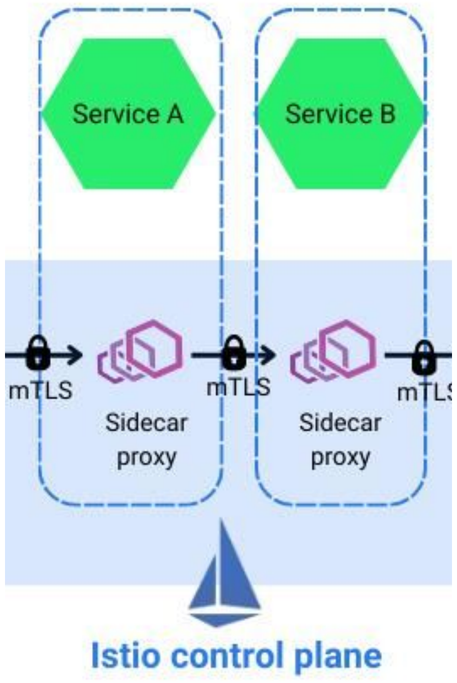
```
{__name__="feedback_no_total", app="app... {__name__="feedback_no_total", app="app...
{__name__="feedback_yes_total", app="ap... {__name__="feedback_yes_total", app="ap...
```

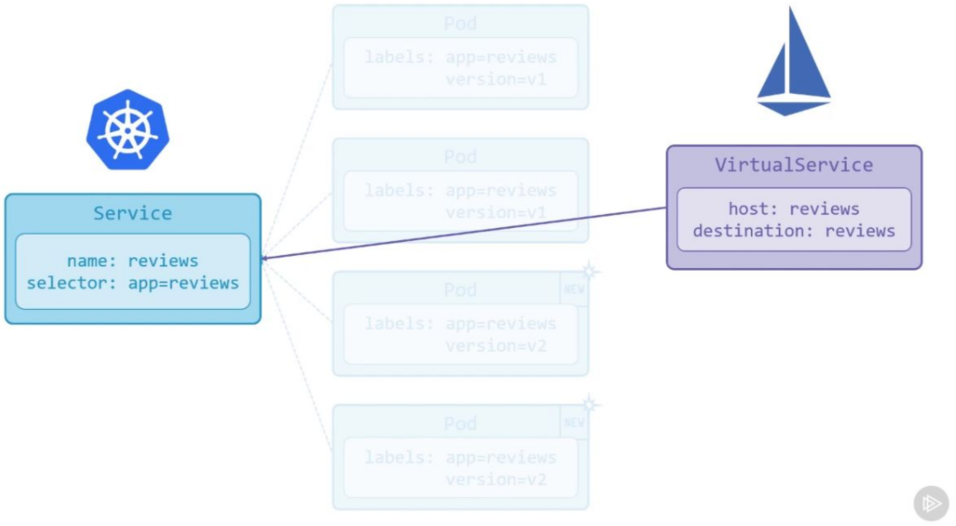
```
{__name__="feedback_no_total", app="app", instance="10.244.0.44:15020", job="kubernetes-pods", namespace="default", node="minikube", poc
{__name__="feedback_no_total", app="app", instance="10.244.0.45:15020", job="kubernetes-pods", namespace="default", node="minikube", poc
```

```
{__name__="feedback_yes_total", app="app", instance="10.244.0.44:15020", job="kubernetes-pods", namespace="default", node="minikube", po
{__name__="feedback_yes_total", app="app", instance="10.244.0.45:15020", job="kubernetes-pods", namespace="default", node="minikube", po
```

```
{__name__="inference_time_histogram_count", app="app", endpoint="/predict", instance="10.244.0.44:15020", job="kubernetes-pods", method="
{__name__="inference_time_histogram_count", app="app", endpoint="/predict", instance="10.244.0.45:15020", job="kubernetes-pods", method="
```

# 5. Experimental Setup: Istio




Service
name: reviews
selector: app=reviews
Pod
labels: app=reviews version=v1
Pod
labels: app=reviews version=v1
Pod
labels: app=reviews version=v2
Pod
labels: app=reviews version=v2
VirtualService
host: reviews
destination: reviews

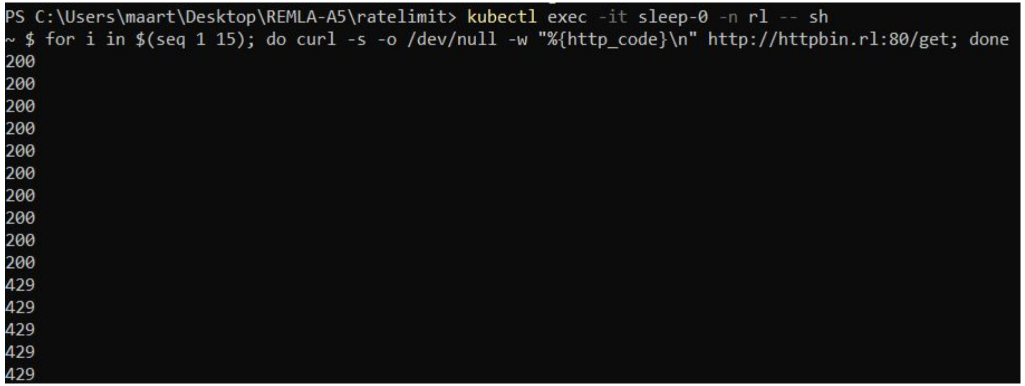
## 6. Additional Istio Use Case-Rate Limiting

**Egress Extension**

In Istio, the Egress extension is used to manage and control traffic to external services in the service grid.

**Adjustments to the Base Deployment**

- Increase ENVOYFILTER configuration
- Enable and verify the flow limit function
- Monitoring and log analysis
- Adjustment and optimization strategy



```
PS C:\Users\maart\Desktop\REMLA-A5\ratelimit> kubectl exec -it sleep-0 -n rl -- sh
~ $ for i in $(seq 1 15); do curl -s -o /dev/null -w "%{http_code}\n" http://httpbin.rl:80/get; done
200
200
200
200
200
200
200
200
200
200
429
429
429
429
429
```

# 7. Extension Proposal- Rollback Strategy

**Benefits:**

- Improve System Stability
- Reduce deployment risk

**Suggestions:**

- Configure Automated Rollback with CI/CD Tools
- Blue-Green Deployment

```
├── app
│   ├── src
│   ├── tests
│   └── Dockerfile
├── model-service
│   ├── src
│   ├── tests
│   └── Dockerfile
├── model-training
│   ├── src
│   ├── tests
│   └── Dockerfile
├── operation
│   ├── src
│   ├── tests
│   └── Dockerfile
├── lib-ml
│   ├── src
│   ├── tests
│   └── Dockerfile
└── lib-version
    ├── src
    ├── tests
    └── Dockerfile
```

```
├── backend
│   ├── model-service
│   │   ├── src
│   │   ├── tests
│   │   ├── Dockerfile
│   │   └── rollback-scripts
│   │       ├── rollback.sh
│   │       └── deploy.sh
│   ├── model-training
│   │   ├── src
│   │   ├── tests
│   │   └── Dockerfile
│   ├── lib-ml
│   │   ├── src
│   │   ├── tests
│   │   └── Dockerfile
│   └── lib-version
│       ├── src
│       ├── tests
│       └── Dockerfile
├── frontend-app
│   ├── src
│   ├── tests
│   └── Dockerfile
├── docs
│   ├── architecture.md
│   ├── api-docs.md
│   └── user-guide.md
└── tests
    ├── unit
    ├── integration
    └── e2e
```

## Q&A

**Thanks for listening! Questions?**