Development and Operation of Distributed Applications

Group 21

Ceylin Ece 5716950
Dibyendu Gupta 5031125
Taeyong Kwon 6283276
Guotao Gou 6380743

Automatic Versioning & Release

Stable Release workflow:

1. Runs only on manual **release** or **workflow** to avoid conflict.

2. Remove the SNAPSHOT, then git tag this version, then publish the release.

3. Then increment the version and then append SNAPSHOT.

```
<groupId>tudelft.doda2025.team21</groupId>
<artifactId>lib-version</artifactId>
<version>1.5.1-SNAPSHOT</version>
```

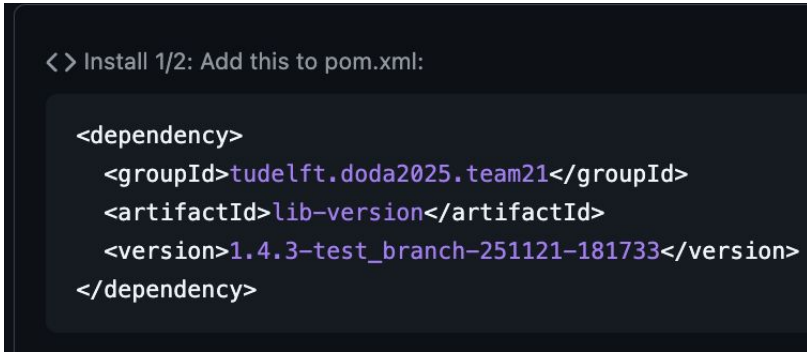
Pom.xml -> source of truth

https://github.com/doda2025-team21/lib-version/tree/main/.github/workflows

Automatic Versioning & Release

Pre Release workflow:

1. Runs only on every push.
2. Remove the SNAPSHOT, then set version as ${BASE_VERSION}-${BRANCH_CLEAN}-${TIMESTAMP}



https://github.com/doda2025-team21/lib-version/tree/main/.github/workflows

Automatic Versioning & Release

```
- name: Determine base version from pom.xml
  id: base_version
  run: |
    # remove -SNAPSHOT to get base version
    mvn versions:set -DremoveSnapshot
    BASE_VERSION=$(mvn -q -Dexpression=project.version -DforceStdout help:evaluate)
    echo "BASE_VERSION=$BASE_VERSION" >> $GITHUB_ENV
    echo "Base version: $BASE_VERSION"

- name: Create pre-release version string
  id: prerelease
  run: |
    # branch name like: feature/new-api -> feature-new-api
    BRANCH_RAW="${GITHUB_REF_NAME}"
    BRANCH_CLEAN=$(echo "$BRANCH_RAW" | tr '/' '-')

    TIMESTAMP=$(date +'%y%m%d-%H%M%S')

    PRE_VERSION="${BASE_VERSION}-${BRANCH_CLEAN}-${TIMESTAMP}"

    echo "Pre-release version: $PRE_VERSION"
    echo "PRE_VERSION=$PRE_VERSION" >> $GITHUB_ENV

- name: Set pom.xml to pre-release version
  run: |
    mvn -B versions:set -DnewVersion=${PRE_VERSION}
    mvn -B versions:commit
```

https://github.com/doda2025-team21/lib-version/tree/main/.github/workflows

Software Reuse in Library

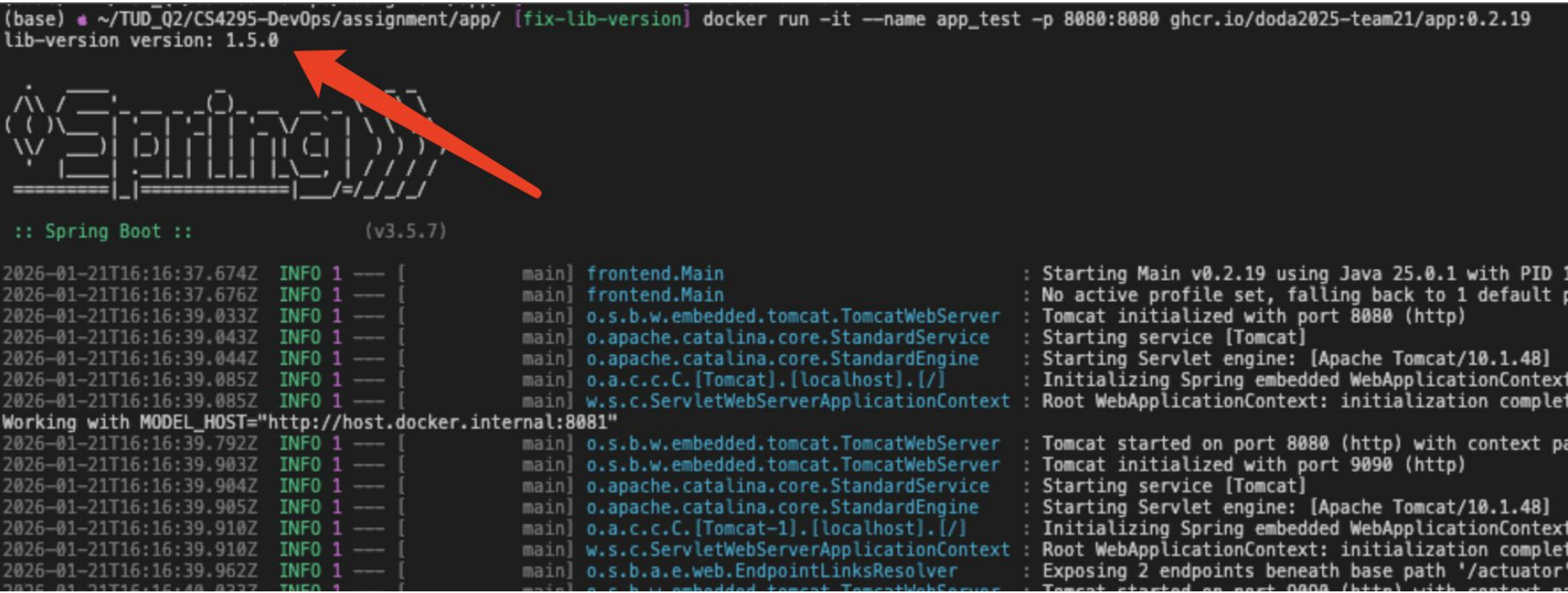
- App depend on stable version lib-version
- App is compiled into **jar** in github workflow. So docker image can directly download it from github, which runs **faster** than local compiling.

```
    <dependency>
        <groupId>tudelft.doda2025.team21</groupId>
        <artifactId>lib-version</artifactId>
        <version>1.5.0</version>
    </dependency>

</dependencies>

<repositories>
    <repository>
        <id>github</id>
        <url>https://maven.pkg.github.com/doda2025-team21/lib-version</url>
    </repository>
</repositories>
```

https://github.com/doda2025-team21/app/blob/main/pom.xml

Software Reuse in Library

- App print out lib-version's **VersionUtil** in main



Container & Orchestration

- Github workflow: use **buildx** to build **multi-architecture**.
- Use **.env** to configure the docker compose deployment.

```yaml
services:
  model-service:
    image: "${MODEL_IMAGE}:${TAG_VERSION:-1.0.5-monitoring}"
    container_name: backend
    ports:
      - "${MODEL_PORT:-8081}:${MODEL_PORT:-8081}"
    environment:
      MODEL_PORT: "${MODEL_PORT:-8081}"
    volumes:
      - ./model:/app/output
    restart: unless-stopped

  app:
    image: "${APP_IMAGE}:${TAG_VERSION:-1.1.0}"
    container_name: frontend
    ports:
      - "${APP_PORT:-8080}:${APP_PORT:-8080}"
    environment:
      APP_PORT: "${APP_PORT:-8080}"
      MODEL_HOST: "http://model-service:${MODEL_PORT:-8081}"
    depends_on:
      - model-service
    restart: unless-stopped
```

Setting up (Virtual) Infrastructure

Provisioning a Kubernetes Cluster

- Kubernetes base platform is reproducibly provisioned.
  - (Vagrant + VirtualBox + Ansible + Flannel CNI + MetalLB + Ingress + Dashboard + Istio)
- Identical cluster on every setup
  - When re-provisioned => same cluster with the same behaviour and capabilities

What the Automation Sets Up

- Vagrant builds:
  - ctrl
  - NUM_WORKERS node-*
  - hosts on a ***host-only network*** (controller at 192.168.56.100, workers starting at 192.168.56.101/102/...).
- Our Vagrantfile generates a valid inventory.cfg for Ansible that contains all (and only) active nodes

Machines are:

- Numbered
- Disposable
- Redundant
- Reproducible

Setting up Software Environment

Idempotent System Configuration

- Ansible playbook uses several built-in modules to achieve an ***idempotent*** provisioning. Some examples include:
  - imports team SSH keys => key is added only if missing
  - disables swap and fstab entries => ensures SWAP stays off
  - loads overlay/br_netfilter => kernel modules ensured loaded
  - enables kubelet => service enabled + running

Setting up Kubernetes

Kubernetes Cluster Setup

Kubernetes Cluster Initialization

- Initialize control plane => advertise address 192.168.56.100 and pod CIDR 10.244.0.0/16 (idempotent via /etc/kubernetes/admin.conf check)
- Distribute kubeconfig => copies kubeconfig to ~/.kube for vagrant and fetches ./kubeconfig for the host
- Join workers automatically => delegates kubeadm token create --print-join-command to the controller and joins each worker if kubelet.conf is missing

Cluster Networking

- Flannel CNI is installed => enables pod-to-pod networking

Kubernetes Platform Add-ons

Platform add-ons

- **Helm** for reproducible add-on deployments
- **MetalLB** with pool **192.168.56.90-192.168.56.99** and readiness **waits**
- Cluster has a working HTTP **Ingress Controller** **=>** ingress-nginx with class nginx and load balancer IP **192.168.56.90**
- Kubernetes **Dashboard** via Helm with an ingress **=> dashboard.local**
- Cluster has a working **Istio Gateway** **=>** Istio 1.25.2 with ingress gateway IP **192.168.56.91**

Helps with

- Monitoring **=>** Ingress, Dashboard
- Traffic Management **=>** Istio

Checking the cluster health from host

```
(base) ceylinece@Ceylins-MacBook-Pro operation % KUBECONFIG=./kubeconfig kubectl get nodes -o wide
KUBECONFIG=./kubeconfig kubectl get pods -A
NAME     STATUS   ROLES           AGE   VERSION   INTERNAL-IP      EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION     CONTAINER-RUNTIME
ctrl     Ready    control-plane   15m   v1.32.4   192.168.56.100   <none>        Ubuntu 24.04.3 LTS   6.8.0-86-generic   containerd://1.7.24
node-1   Ready    <none>          14m   v1.32.4   192.168.56.101   <none>        Ubuntu 24.04.3 LTS   6.8.0-86-generic   containerd://1.7.24
node-2   Ready    <none>          14m   v1.32.4   192.168.56.102   <none>        Ubuntu 24.04.3 LTS   6.8.0-86-generic   containerd://1.7.24
NAMESPACE              NAME                                                READY   STATUS    RESTARTS   AGE
ingress-nginx          ingress-nginx-controller-5b57587585-4m2hr           1/1     Running   0          13m
istio-system           istio-ingressgateway-7cff6d9468-t7xrw               1/1     Running   0          10m
istio-system           istiod-867f8c9f4f-7gr6c                             1/1     Running   0          10m
kube-flannel           kube-flannel-ds-4bpt7                               1/1     Running   0          14m
kube-flannel           kube-flannel-ds-lpgst                               1/1     Running   0          14m
kube-flannel           kube-flannel-ds-wp49m                               1/1     Running   0          15m
kube-system            coredns-668d6bf9bc-2ng5j                            1/1     Running   0          15m
kube-system            coredns-668d6bf9bc-p59m7                            1/1     Running   0          15m
kube-system            etcd-ctrl                                           1/1     Running   0          15m
kube-system            kube-apiserver-ctrl                                 1/1     Running   0          15m
kube-system            kube-controller-manager-ctrl                        1/1     Running   0          15m
kube-system            kube-proxy-b67r2                                    1/1     Running   0          14m
kube-system            kube-proxy-rqjlb                                    1/1     Running   0          15m
kube-system            kube-proxy-wcf5s                                    1/1     Running   0          14m
kube-system            kube-scheduler-ctrl                                 1/1     Running   0          15m
kubernetes-dashboard   kubernetes-dashboard-api-5b57c4d774-mtsds           1/1     Running   0          8m6s
kubernetes-dashboard   kubernetes-dashboard-auth-78f9f9c85c-n7jkm          1/1     Running   0          8m6s
kubernetes-dashboard   kubernetes-dashboard-kong-9b49bb989-55282           1/1     Running   0          12m
kubernetes-dashboard   kubernetes-dashboard-metrics-scraper-74d8cb664-fbk49   1/1     Running   0          12m
kubernetes-dashboard   kubernetes-dashboard-web-59b8766cc-7czm5            1/1     Running   0          12m
metallb-system         controller-bb5f47665-w2mp5                          1/1     Running   0          14m
metallb-system         speaker-6d765                                       1/1     Running   0          14m
metallb-system         speaker-6xn24                                       1/1     Running   0          14m
metallb-system         speaker-dzr4l                                       1/1     Running   0          14m
```

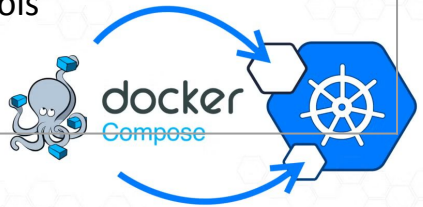
Verifying load-balancers / ingress

```
(base) ceylinece@Ceylins-MacBook-Pro operation % KUBECONFIG=./kubeconfig kubectl get svc -n metallb-system
KUBECONFIG=./kubeconfig kubectl get svc -n ingress-nginx ingress-nginx-controller -o wide
KUBECONFIG=./kubeconfig kubectl get svc -n kubernetes-dashboard
KUBECONFIG=./kubeconfig kubectl get svc -n istio-system istio-ingressgateway -o wide
NAME                      TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)   AGE
metallb-webhook-service   ClusterIP   10.109.229.130   <none>        443/TCP   15m
NAME                       TYPE           CLUSTER-IP     EXTERNAL-IP     PORT(S)                      AGE   SELECTOR
ingress-nginx-controller   LoadBalancer   10.103.56.14   192.168.56.90   80:31662/TCP,443:30903/TCP   14m   app.kubernetes.io/component=controller,app.kubernetes.io/instance=ingres
s-nginx,app.kubernetes.io/name=ingress-nginx
NAME                                    TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)    AGE
kubernetes-dashboard-api                ClusterIP   10.105.177.118   <none>        8000/TCP   13m
kubernetes-dashboard-auth               ClusterIP   10.103.51.194    <none>        8000/TCP   13m
kubernetes-dashboard-kong-proxy         ClusterIP   10.110.242.179   <none>        443/TCP    13m
kubernetes-dashboard-metrics-scraper    ClusterIP   10.99.56.243     <none>        8000/TCP   13m
kubernetes-dashboard-web                ClusterIP   10.96.141.36     <none>        8000/TCP   13m
NAME                   TYPE           CLUSTER-IP      EXTERNAL-IP     PORT(S)                                      AGE   SELECTOR
istio-ingressgateway   LoadBalancer   10.101.80.148   192.168.56.91   15021:31937/TCP,80:32693/TCP,443:31725/TCP   11m   app=istio-ingressgateway,istio=ingressgateway
```

Kubernetes Usage

Why are we moving from Docker-Compose ⇨Kubernetes?

Docker Limitations	How Kubernetes addresses them?
Single-host constraint	Runs across cluster of machines; better for scalability and high availability
Basic Load-Balancing	Sophisticated load-balancing across pods in a cluster: - defining routing rules, - configurable for canary and shadow releases - DNS-based service discovery
No Self-healing	Kubernetes Scheduler and Controller Manager perform automatic container restarts, and rescheduling on pod failure.
App Monitoring	Allows exposing metrics that can be consumed by other tools (Prometheus) and perform visualizations (Grafana) and alerting (alertManager).



How sms-checker Is Deployed on Kubernetes Clusters

- Deployed using Helm-chart
  - Package manager that describe Kubernetes resources
- Resources:
  - Deployments: Creates pods with specific labels.
  - Services: Allows for internal communication through Cluster IP. (accessible only within the cluster)
  - ServiceMonitor: Automatically discovers and scrapes metrics from K8 services. Bridges Prometheus and application.

App Monitoring


Grafana
Prometheus
Alertmanager

Monitoring the Cluster

What are we monitoring?

- Kubernetes health
- App frontend metrics

How?

- Endpoints are exposed through app-frontend endpoint /metrics
- Prometheus scrapes the Service for metrics via ServiceMonitors
- Grafana queries Prometheus to visualize data
- AlertManager receives alerts from Prometheus: when Prometheus rule is Firing
- AlertManager sends alerts to configured receivers (through E-mails, Slack, etc.)

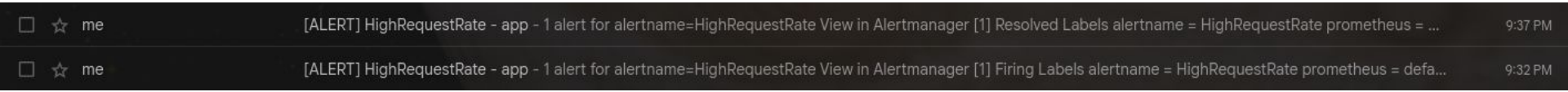
Relevance?

- Replicates production systems
- Infrastructure-as-code: managing and monitoring resources

Alerting via Email

- Enabled and configured Alertmanager with SMTP-based email notifications
  - Triggers when traffic exceeds 2 requests/min for 1 minutes
- Injected SMTP credentials securely at install time via Helm values ; possible improvement using “Secrets”
- Verified alert lifecycle:
  Inactive → Pending → Firing → Resolved
- Integrated Grafana dashboards for real-time metric visualization

```yaml
kind: PrometheusRule
metadata:
  name: high-traffic-alert
  labels:
    release: {{ .Release.Name }}
spec:
  groups:
    - name: traffic-alerts
      rules:
        - alert: HighRequestRate
          expr: sum(rate(sms_requests_total[1m])) * 60 > 2
          for: 1m
          labels:
            severity: warning
```




Kubernetes Deployment
Monitor > Applications > Kubernetes Deployment
Dashboard Add Dashboards Display Help
Deployment
php-apache
Namespace
default
Condition
Summary All conditions OK
CPU usage
2.50
1.67
0
Memory usage
128.00 MB
92.56 MB
0
CPU resources: Deployment
3.00
2.50
2.00
1.50
1.00
0.50
0
13:30 13:40 13:50 14:00 14:10
Memory resources: Deployment
112 MB
96 MB
80 MB
16 MB
0
13:30 13:40 13:50 14:00 14:10
CPU usage: Pods
0.60
0.40
0.20
0
13:30 13:45 14:00 14:15
Mean Usage
Median Usage
Usage by host
Memory usage: Pods
32 MB
24 MB
16 MB
8 MB
0
13:30 13:45 14:00 14:15
Mean Usage
Median Usage
Usage by host
Grafana
Desired replicas
6
0
Up-to-date replicas
6
0
Ready replicas
5
0
Service: CRIT
Pod resources: Deployment
6
4
2
0
13:30 13:40 13:50 14:00 14:10
Pods
Deployment problems
State Service Object Name Summary
CRIT Status Object Pod Name php-apache-6cb4d4d74b-m6x8g Pending: since 3 hours 12 minutes (warn/crit at 5 minutes 0 seconds/10 minutes 0 seconds)
CRIT Replicas Object Deployment Name php-apache Ready: 5/6, Up-to-date: 6/6, Not ready for: 3 hours 17 minutes (warn/crit at 5 minutes 0 seconds/10 minutes 0 seconds) CRIT

Grafana Dashboard and Metrics

Metrics tracked: sms_request_total

<insert dashboard image>

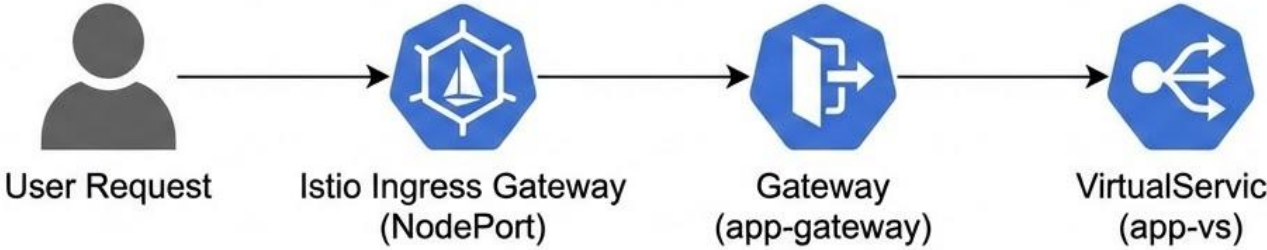
Key deliverables of A4

Key deliverables:

Traffic Management: Gateway, VirtualService, and DestinationRules

- **Canary Release:** Weighted routing (90% Stable / 10% Canary)
- **Sticky Sessions:** Consistent hashing via cookies

Traffic Management Architecture



Routing Logic

- The VirtualService splits traffic based on headers (x-canary) or weights.
- Traffic is directed to specific DestinationRules which define the subsets: Stable: Version 0.2.17 (90% traffic). Canary: Version 0.2.14 (10% traffic).
- This ensures consistent routing between the app and model services (e.g., old-to-old, new-to-new).

Canary Deployment Strategy

Configuration:

- Defined in values.yaml under istio.canary
- **Default Route:** 90% weight to Stable, 10% weight to Canary
- **Header Override:** Requests with header x-canary: true are forced to the canary subset for testing purposes

Note: This fulfills the A4 requirement to "use DestinationRules for a canary release to a small fraction of the users" + 90/10 traffic

```
taea@taea-HP-Laptop-15-fd0xxx:~/doda-assignment/operation$
for i in {1..10}; do
  curl -s http://sms-istio.local/sms/ -X POST -H "Content-Type: application/json" -d '{"sms": "test '$i'"}' > /dev/null
  sleep 1
done

echo "=== STABLE ===" && KUBECONFIG=./kubeconfig kubectl logs deploy/app-stable -c app --tail=20 | grep "test "
echo "=== CANARY ===" && KUBECONFIG=./kubeconfig kubectl logs deploy/app-canary -c app --tail=20 | grep "test "
=== STABLE ===
Found 2 pods, using pod/app-stable-8447559955-p54mp
Requesting prediction for "test message 1" ...
Requesting prediction for "test message 2" ...
Requesting prediction for "test message 5" ...
Requesting prediction for "test 4" ...
Requesting prediction for "test 5" ...
Requesting prediction for "test 4" ...
=== CANARY ===
Requesting prediction for "test 6" ...
```

Sticky Sessions (Stability)

The Problem: Ensuring users stay on the same version during an experiment session.

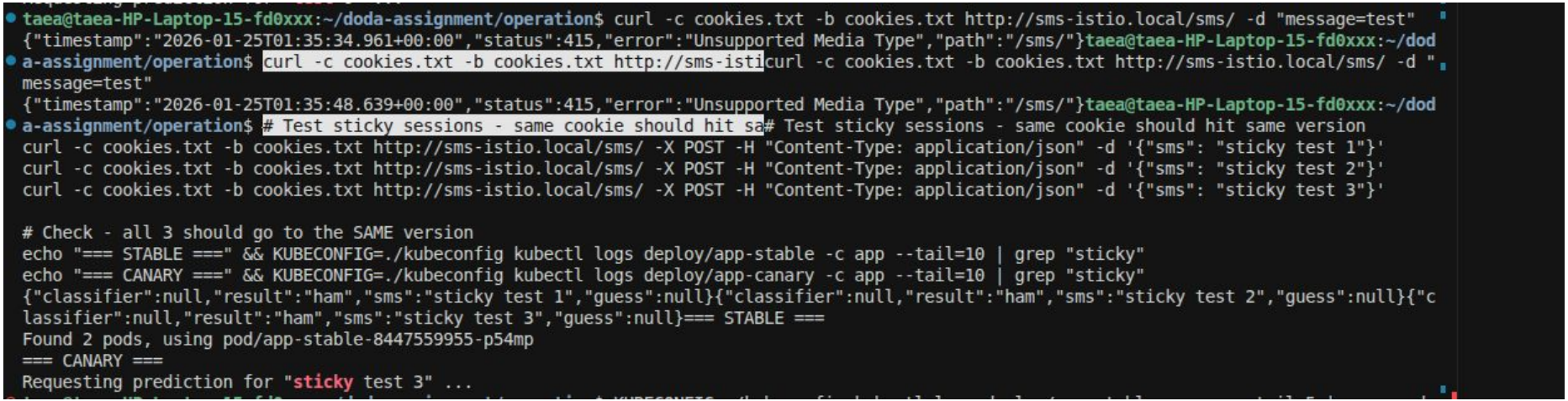
The Solution: HTTP Cookie-based consistent hashing.

Implementation:

- Cookie Name: sms-session.
- TTL: 3600 seconds (1 hour).

How it works:

1. First request: Istio routes based on the 90/10 weight.
2. Response: Includes Set-Cookie: sms-session=….
3. Subsequent requests: The cookie forces the user to the same subset (Stable or Canary).


taea@taea-HP-Laptop-15-fd0xxx:~/doda-assignment/operation$ curl -c cookies.txt -b cookies.txt http://sms-istio.local/sms/ -d "message=test"
{"timestamp":"2026-01-25T01:35:34.961+00:00","status":415,"error":"Unsupported Media Type","path":"/sms/"}taea@taea-HP-Laptop-15-fd0xxx:~/dod
a-assignment/operation$ curl -c cookies.txt -b cookies.txt http://sms-isticurl -c cookies.txt -b cookies.txt http://sms-istio.local/sms/ -d "
message=test"
{"timestamp":"2026-01-25T01:35:48.639+00:00","status":415,"error":"Unsupported Media Type","path":"/sms/"}taea@taea-HP-Laptop-15-fd0xxx:~/dod
a-assignment/operation$ # Test sticky sessions - same cookie should hit sa# Test sticky sessions - same cookie should hit same version
curl -c cookies.txt -b cookies.txt http://sms-istio.local/sms/ -X POST -H "Content-Type: application/json" -d '{"sms": "sticky test 1"}'
curl -c cookies.txt -b cookies.txt http://sms-istio.local/sms/ -X POST -H "Content-Type: application/json" -d '{"sms": "sticky test 2"}'
curl -c cookies.txt -b cookies.txt http://sms-istio.local/sms/ -X POST -H "Content-Type: application/json" -d '{"sms": "sticky test 3"}'
Check - all 3 should go to the SAME version
echo "=== STABLE ===" && KUBECONFIG=./kubeconfig kubectl logs deploy/app-stable -c app --tail=10 | grep "sticky"
echo "=== CANARY ===" && KUBECONFIG=./kubeconfig kubectl logs deploy/app-canary -c app --tail=10 | grep "sticky"
{"classifier":null,"result":"ham","sms":"sticky test 1","guess":null}{"classifier":null,"result":"ham","sms":"sticky test 2","guess":null}{"c
lassifier":null,"result":"ham","sms":"sticky test 3","guess":null}=== STABLE ===
Found 2 pods, using pod/app-stable-8447559955-p54mp
=== CANARY ===
Requesting prediction for "sticky test 3" ...

Verification: Rate Limiting Enforcement

Normal Traffic

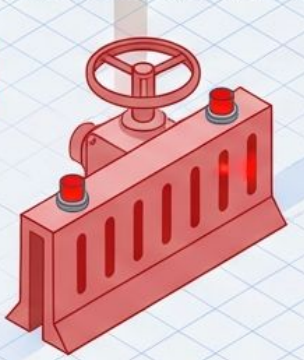
Requests 1-10 (Within Burst Limit)
Request 1: HTTP 200 OK
Request 2: HTTP 200 OK
Request 3: HTTP 200 OK
...
Request 10: HTTP 200 OK

Blocked Traffic

Request 11 (Limit Exceeded)

HTTP 429 Too Many Requests

x-local-rate-limit: TOO_MANY_REQUESTS



Continuous Experimentation

- **The Experiment:** Testing a new frontend UI design in the canary version (0.2.14)
- **Hypothesis:** The new UI will improve user engagement
- **Observability:**
  - **Grafana Dashboard:** decision-making-a4.json performs an A/B comparison
  - **Metrics Tracked:** Latency distribution, Request Rate, and Error rate
- **Decision Process:** Compare stable vs. canary metrics in the dashboard to accept or reject the release.

Configuration & Deployment

Helm Integration:

- All Istio features are configurable via values.yaml (e.g.istio.enabled, canary.enabled, stickySession.enabled)
- **Ingress Name:** Configurable via ingressGateway.selector to support different clusters, meeting the assignment requirement for portability

- **Deployment Commands:**
  - Namespace labeled for Istio injection: kubectl label namespace default istio-injection=enabled
  - Deployed via Helm upgrade command

Want to understand the flow completely?

docs/deployment.md: Explains deployment structure and data flow
docs/extension.md: Extension proposal for release engineering improvements
docs/continuous-experimentation.md: Details the hypothesis and decision metrics