KubeCon CloudNativeCon

THE LINUX FOUNDATION

OPEN SOURCE SUMMIT

AI_dev

Open Source GenAI & ML Summit

China 2024

KubeCon

CloudNativeCon

THE LINUX FOUNDATION

OPEN SOURCE SUMMIT

AI_dev

Open Source GenAI & ML Summit

China 2024

Rollout Patterns: Smoothly Migrate and Roll Out Your Microservices

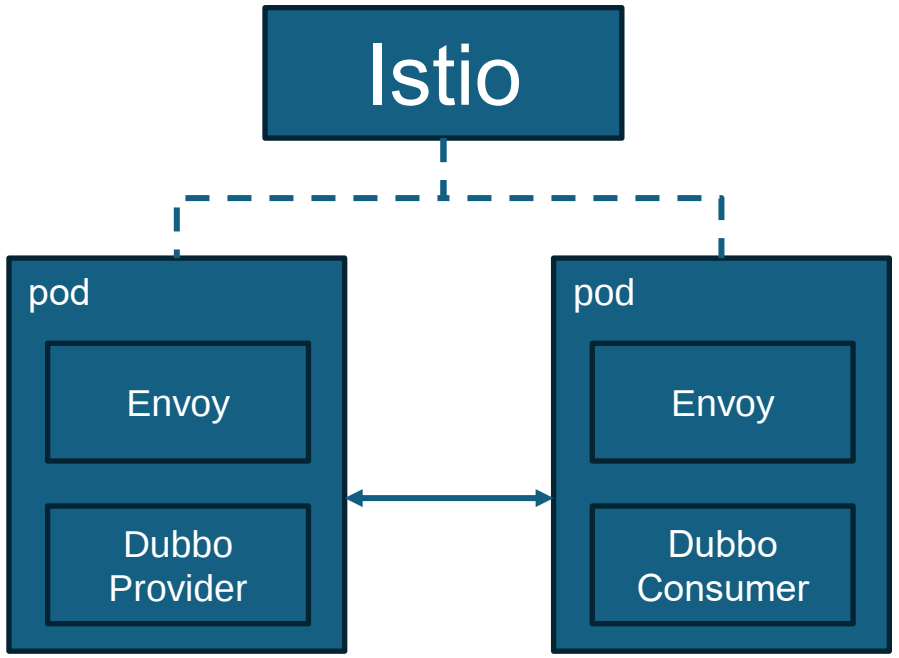
Tim Xiao, DaoCloud

About me

- Tim @muma378
- DaoCloud, Software Engineer, DevOps Platform Leader
- Argo Project Contributor
- Hiking 🏔️、Badminton 🏸 and Coffee ☕ Lover

Background

- Services were built with Java and Dubbo2
- Microservices architecture
- Migrate to service mesh
- Want canary rollout



Rollout Patterns

- PATTERN 1: One service at a time
- Usually happen for monolithic or compatible interfaces
- The delivery order could be very important
- Low requirement on Ops but high requirement on Dev

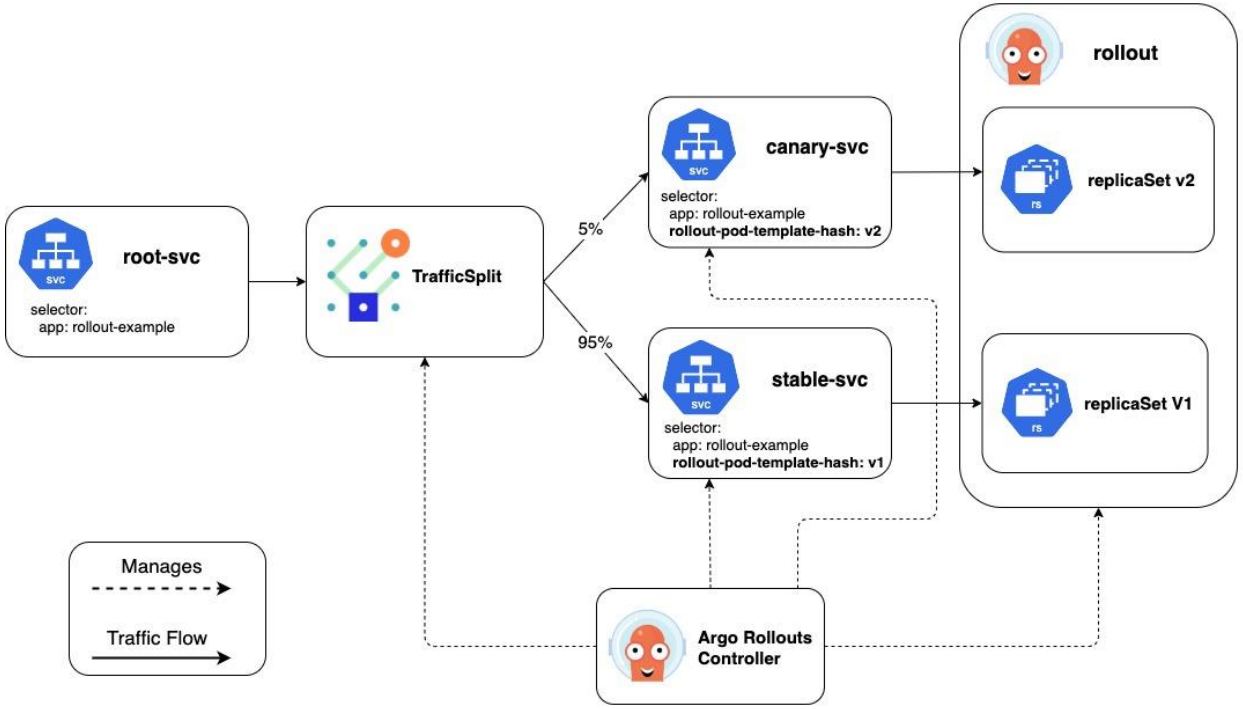
How you deliver your services?

Rollout Patterns

- The ideal scenario for argo-rollouts

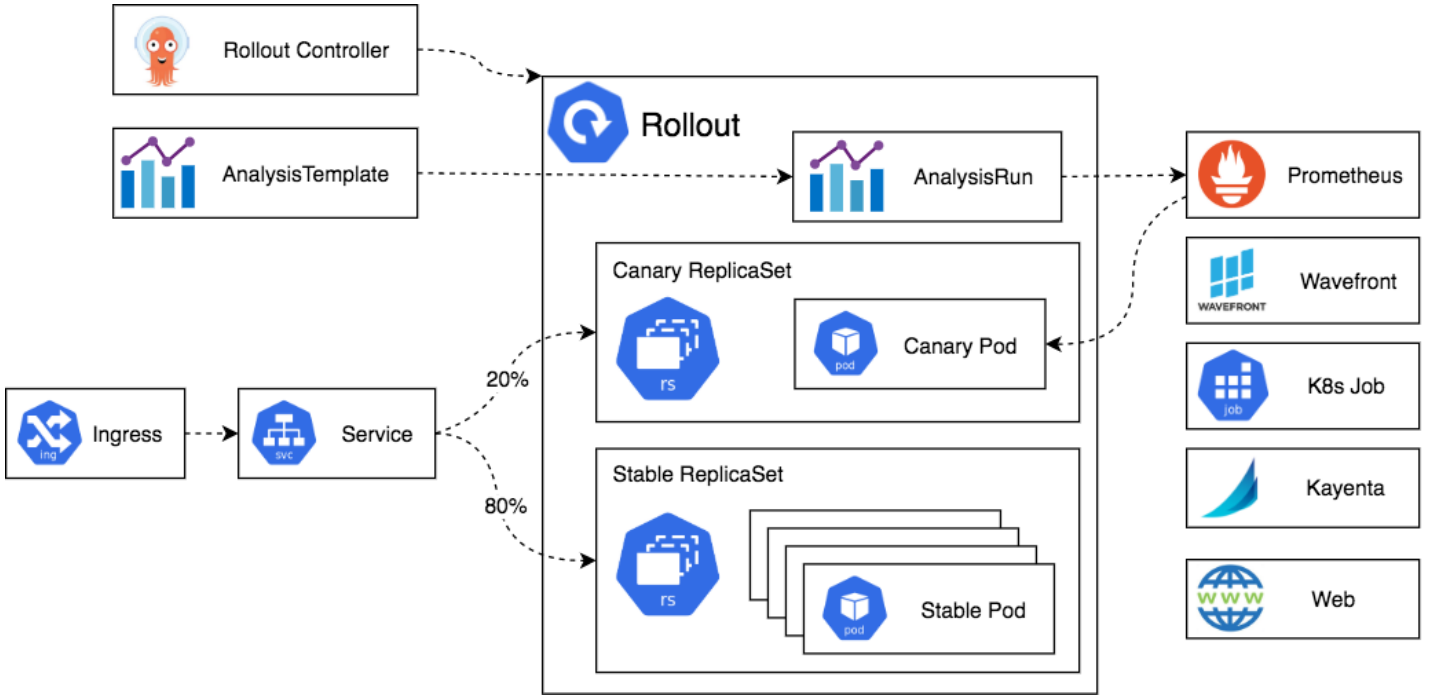
What is Argo-Rollouts

- (optionally) integrates with ingress controllers and service meshes, leveraging their traffic shaping abilities to gradually shift traffic to the new version during an update

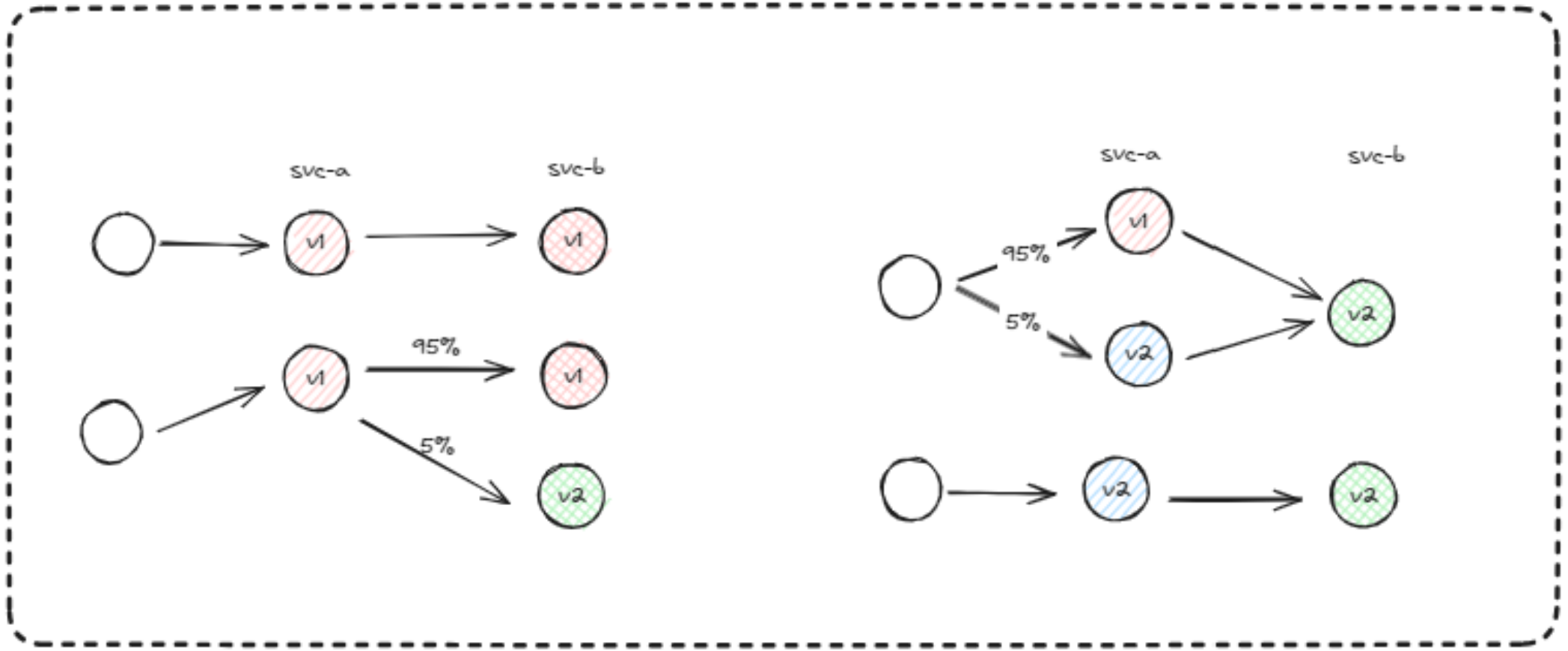


What is Argo-Rollouts

- can query and interpret metrics from various providers to verify key KPIs and drive automated promotion or rollback during an update



Pattern 1

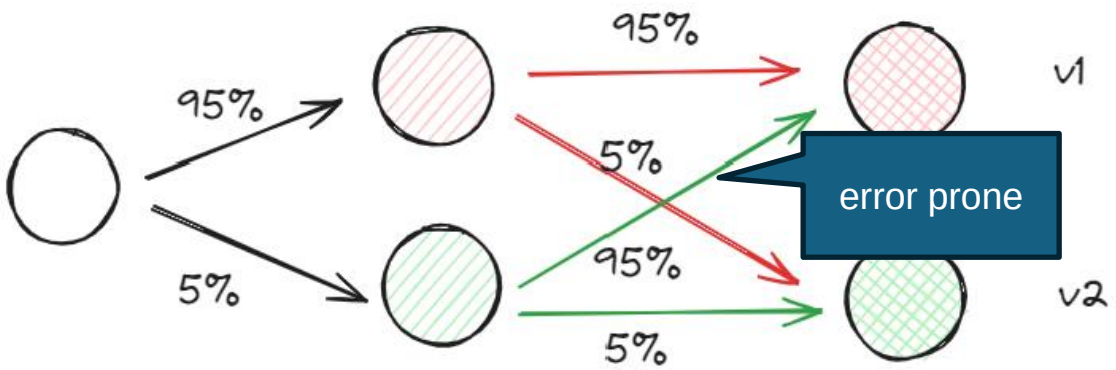


However...

- When a change involves multiple components, they are preferred to be delivered in together
- Not only will the workloads be updated, but the configuration policies will also change

Rollout Patterns

- PATTERN 2: multiple services, each is backward-compatible
- Compared with pattern 1, applications are delivered as a whole
- Backward-compatible is for callers, not for callees
- Traffic become very complicate, hard to troubleshoot, hard to rollback

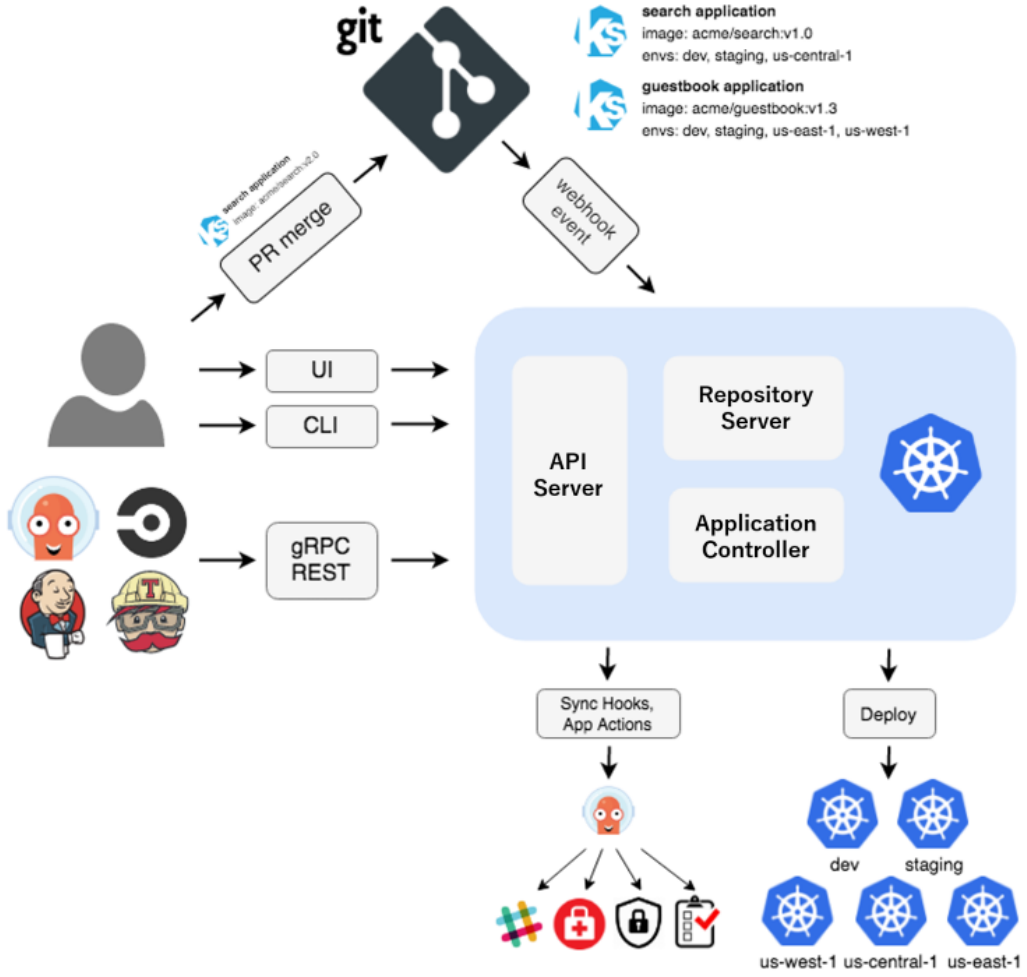


All you need is a delivery tool to control the order of deploying

Argo CD

What is Argo CD

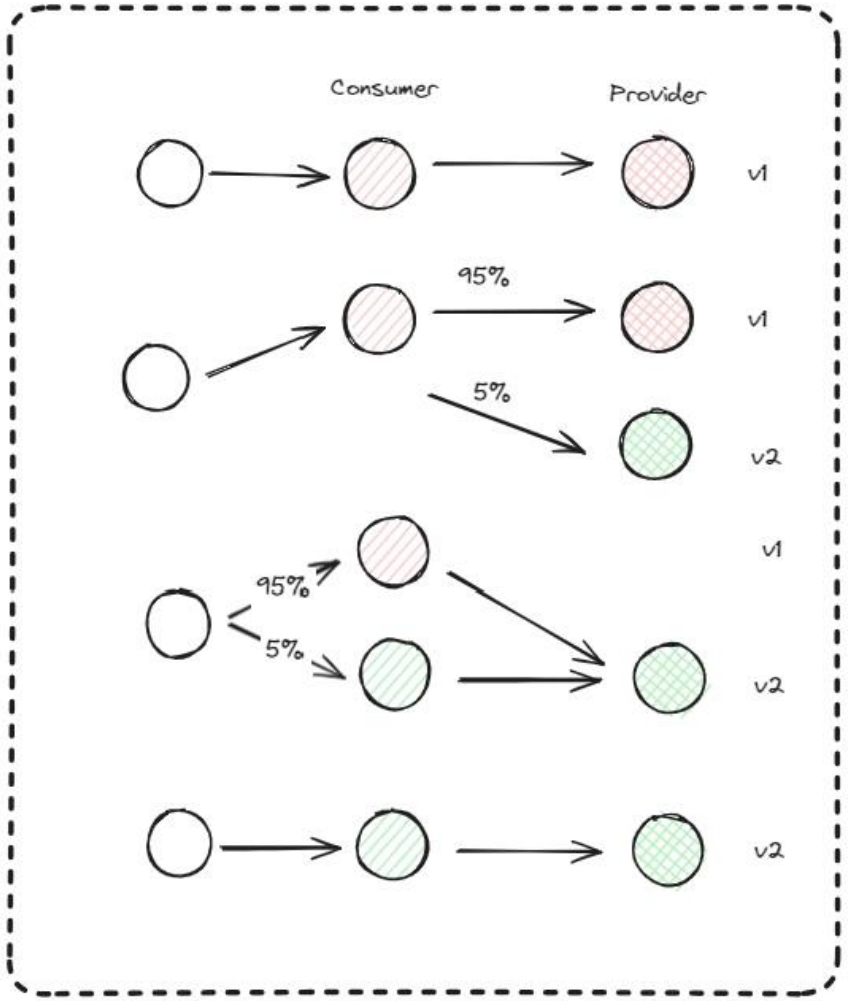
- Argo CD follows the **GitOps** pattern of using Git repositories as the source of truth for defining the desired application state
- Argo CD automates the deployment of the desired application states in the specified target environments
- Argo CD reports & visualizes the differences, while providing facilities to automatically or manually sync the live state back to the desired target state.
- Support to define waves to apply resource

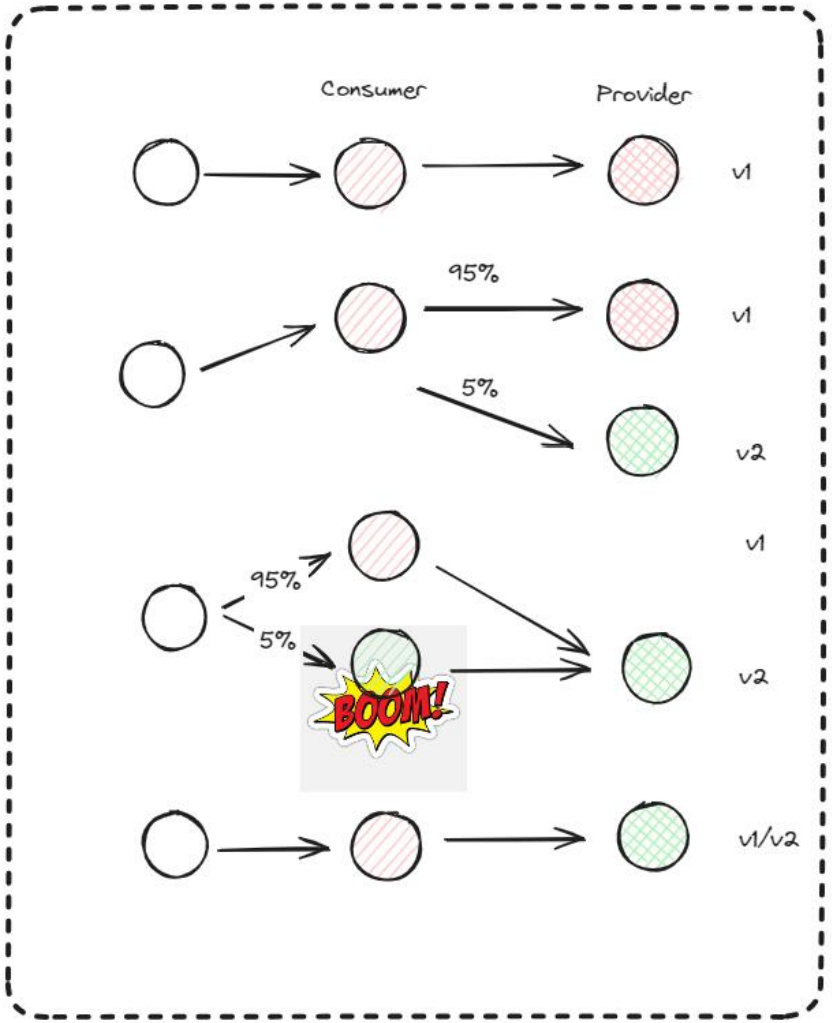


Pattern 2

1. Identify the dependencies between services and design the order according to the direction
2. Progress one by one and monitor traffic shift
3. Rollback only the releasing service once outage occurs

Pattern 2

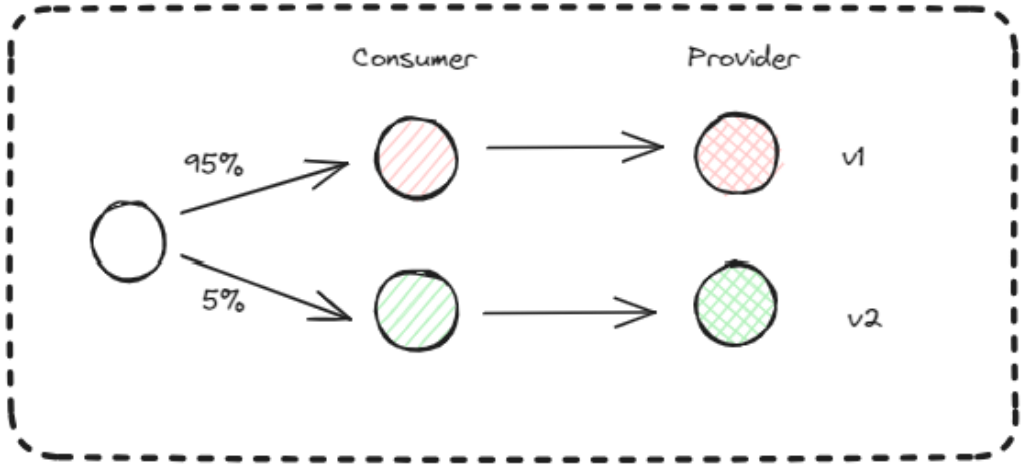

Consumer
Provider
v1
95%
v1
5%
v2
v1
95%
5%
v2
v2


Consumer
Provider
v1
95%
v1
5%
v2
v1
95%
5%
BOOM!
v2
v1/v2

LIVE DEMO

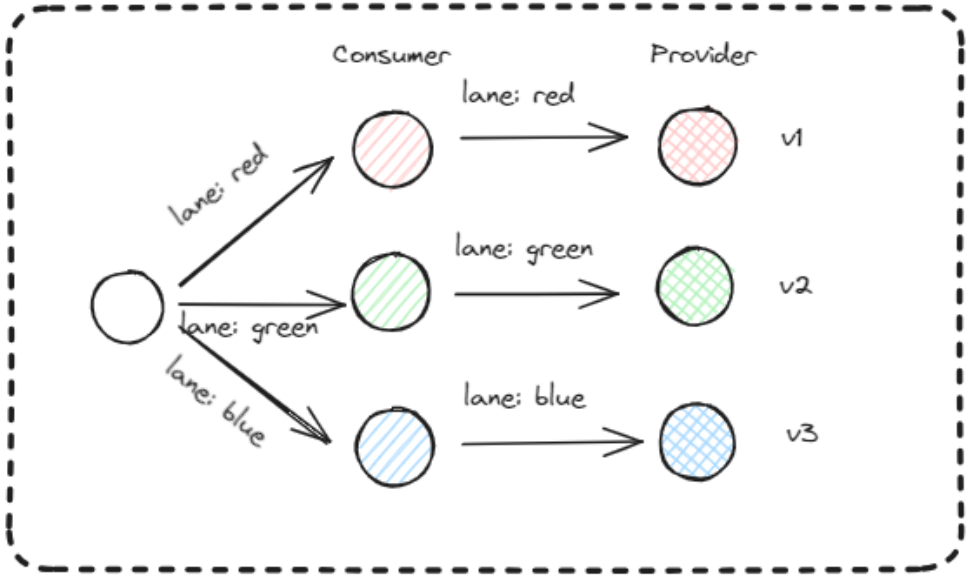
Rollout Patterns

- PATTERN 3: multiple services with versions lock
- Compared with pattern 2, APIs are not compatible, requests are passed alone the lane, lane-cross is not permitted
- Rollout means shifting traffic in the entrance



First of all

- How to control the endpoint which consumer connects to?
- Headers (aka. A/B Testing)
- A non-invade solution?
- Auto Inject headers via OpenTelemetry and WasmPlugin



Pattern 3

1. Deploy all services but don’t update traffic rules, no traffic will be directed to the new release at present
2. Shift traffic to the new release
3. Progressive promote till the new release take over all traffic
4. Scale the old replicas to 0
5. Reset the rules for the next rollout

More details

How to update replicas but no update route?

setCanaryScale

```
            port: 20880
  strategy:
    canary:
      canaryService: dubbo3-provider-canary
      stableService: dubbo3-provider
      trafficRouting:
        managedRoutes:
          - name: test-v2
        istio:
          virtualService:
            name: provider
            routes:
              - test-v1
      steps:
        - setCanaryScale:
            weight: 100
        - setHeaderRoute:
            name: test-v2
            match:
            - headerName: lane
              headerValue:
                exact: test-v2
        - pause: {}
```

More details

How to add different headers to requests forwarded to different destination?

ask traffic management tools for help

```yaml
apiVersion: networking.istio.io/v1beta1
kind: VirtualService
metadata:
  name: consumer
spec:
  gateways:
    - dubbo
  hosts:
    - "*"
  http:
    - name: test-v1
      route:
        - destination:
            host: dubbo3-consumer
            port:
              number: 9090
          weight: 100
          headers:
            request:
                add:
                  lane: test-v1
        - destination:
            host: dubbo3-consumer-canary
            port:
              number: 9090
          weight: 0
          headers:
            request:
              add:
                lane: test-v2
```

add lane:test-v1

add lane:test-v2

More details

How to update route to only accept requests with specified headers

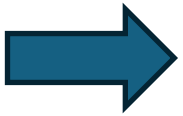
setHeaderRoute

```
            port: 20880
strategy:
  canary:
    canaryService: dubbo3-provider-canary
    stableService: dubbo3-provider
    trafficRouting:
      managedRoutes:
        - name: test-v2
      istio:
        virtualService:
          name: provider
          routes:
            - test-v1
    steps:
      - setCanaryScale:
          weight: 100
      - setHeaderRoute:
          name: test-v2
          match:
          - headerName: lane
            headerValue:
              exact: test-v2
      - pause: {}
```

add new header match route

More details


```yaml
apiVersion: networking.istio.io/v1beta1
kind: VirtualService
metadata:
  name: provider
spec:
  gateways:
    - mesh
  hosts:
    - dubbo3-provider
  http:
    - name: test-v1
      match:
        - headers:
            lane:
              exact: test-v1
      route:
        - destination:
            host: dubbo3-provider
            port:
              number: 20880
          weight: 100
    - route:
        - destination:
            host: dubbo3-provider
            port:
              number: 20880
```

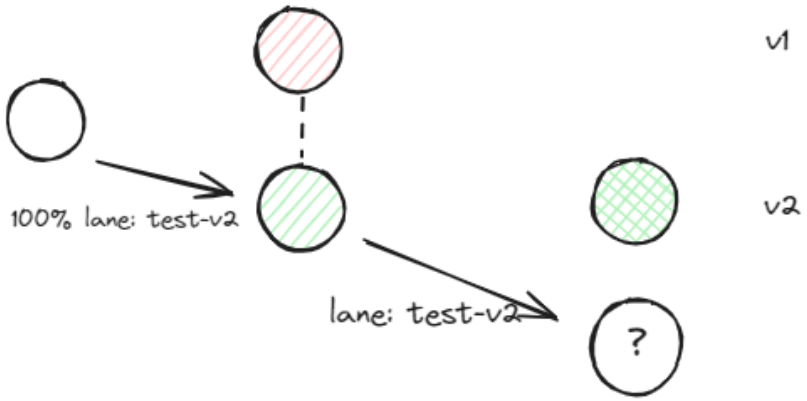


```yaml
  http:
    - name: test-v1
      match:
        - headers:
            lane:
              exact: test-v1
      route:
        - destination:
            host: dubbo3-provider
            port:
              number: 20880
          weight: 100
    - name: test-v2
      match:
        - headers:
            lane:
              exact: test-v2
      route:
        - destination:
            host: dubbo3-provider-canary
            port:
              number: 20880
          weight: 100
    - route:
        - destination:
            host: dubbo3-provider
            port:
              number: 20880
```

More details

Once a rollout was finished, header route will be cleared, but requests are still with headers (because the entrance rollout is not finished yet), how to handle this?

add default route



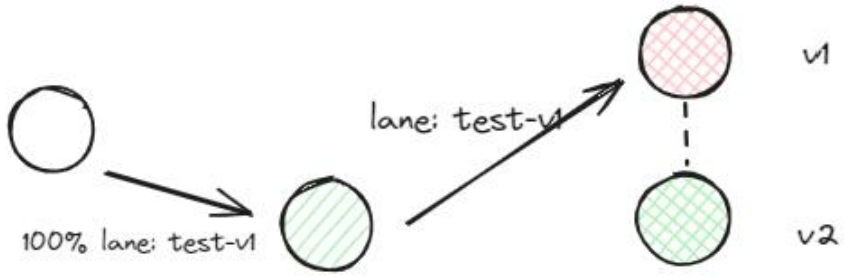

```yaml
apiVersion: networking.istio.io/v1beta1
kind: VirtualService
metadata:
  name: provider
spec:
  gateways:
    - mesh
  hosts:
    - dubbo3-provider
  http:
    - name: test-v1
      match:
        - headers:
            lane:
              exact: test-v1
      route:
        - destination:
            host: dubbo3-provider
            port:
              number: 20880
          weight: 100
    - route:
        - destination:
            host: dubbo3-provider
            port:
              number: 20880
```

route if no match

More details

Why do we have to leave the entrance rollout finish last?

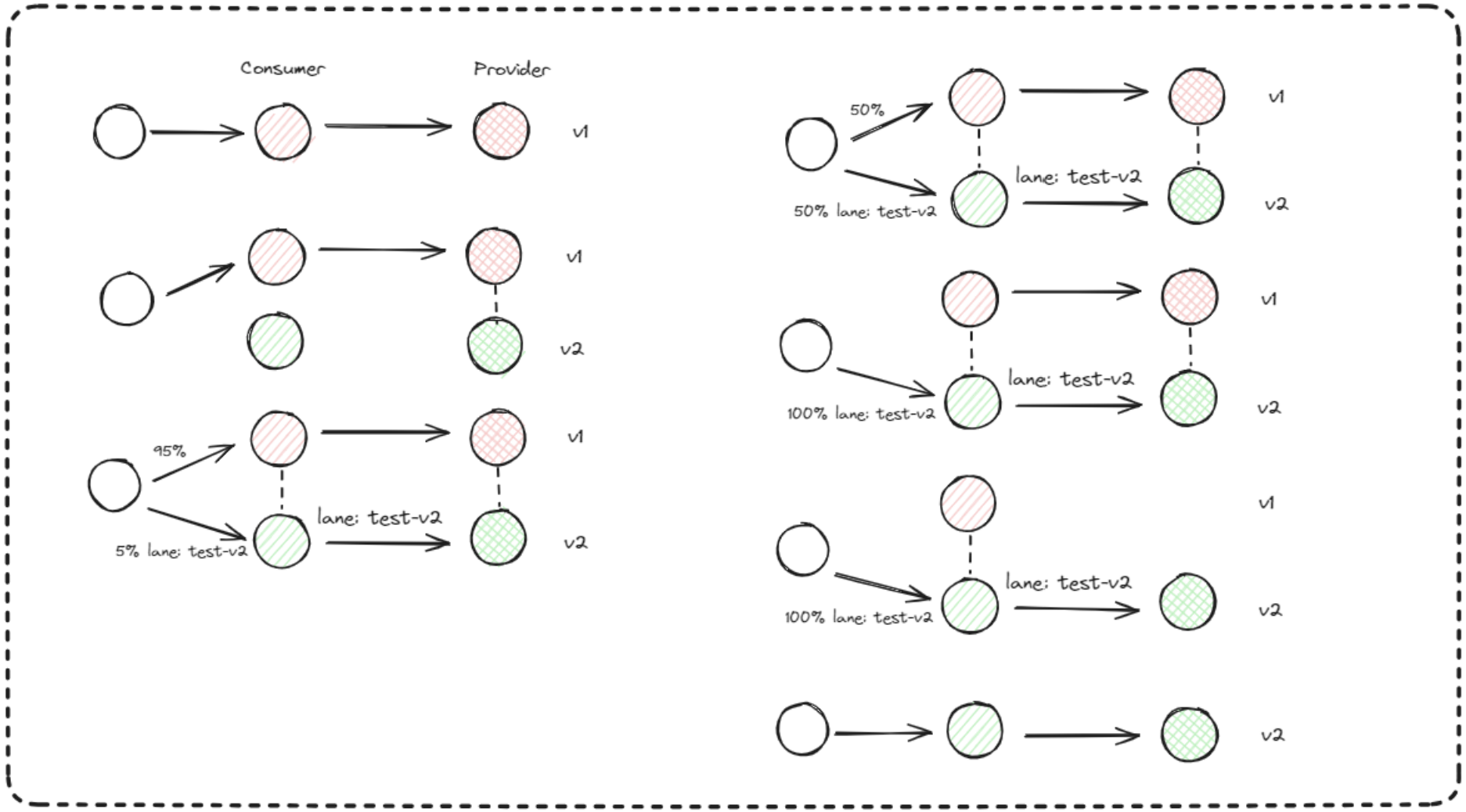
If we don’t, the old stable service will be scaled down to 0 and the canary service will be the new stable. Which means all traffic will go the lane v1




```
29  strategy:
30    canary:
31      canaryService: dubbo3-consumer-canary
32      stableService: dubbo3-consumer
33      trafficRouting:
34        istio:
35          virtualService:
36            name: consumer
37            routes:
38              - test-v1
39      steps:
40        - setCanaryScale:
41            weight: 20
42        - pause: {}
43        - setWeight: 20
44        - pause:
45            duration: 10m
46        - setWeight: 50
47        - pause:
48            duration: 10m
49        - setWeight: 100
50        - pause: {}
```

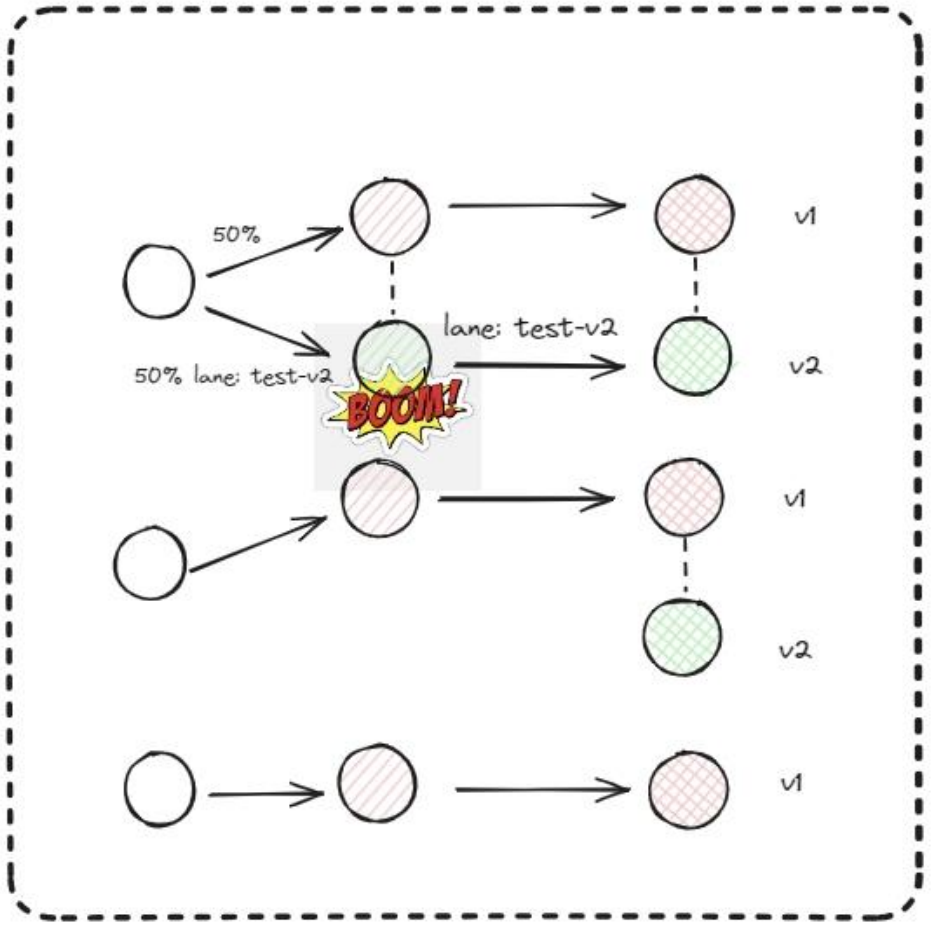
wait here

Pattern 3


Consumer
Provider
v1
v1
v2
95%
5% lane: test-v2
lane: test-v2
v1
v2
50%
50% lane: test-v2
lane: test-v2
v1
v2
100% lane: test-v2
lane: test-v2
v1
v2
100% lane: test-v2
lane: test-v2
v1
v2
v2

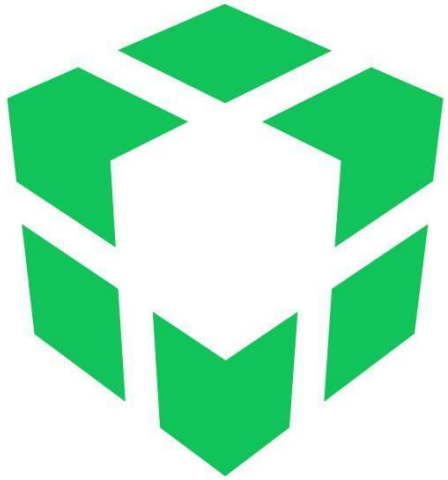
Pattern 3: rollback

- Once an outage happens, abort the entrance service at first, then the others



LIVE DEMO

Questions?



DaoCloud

Appreciate any feedback

wechat @muma378
github @muma378