weaveworks

Flagger

A Progressive Delivery Operator for Kubernetes

Stefan Prodan @stefanprodan

CDF - April 2020

What is Progressive Delivery?

Progressive delivery techniques like Canary Releases and A/B Testing are used to **reduce the risk of introducing a new software version in production** by giving app developers and SRE teams a fine-grained control over the blast radius.

Ingredients:

- CI pipeline that produces immutable build artifacts
- CD pipeline designed for desired state reconciliation
- Smart routing for user facing apps and service to service communication
- Observability (performance stats + business metrics)

Flagger - 2018 Goals

- Give developers confidence in automating the production releases
  - Have control over the blast radius
  - Define the validation process with KPIs and thresholds
  - Extend the validation with automated testing
  - Manual gating for critical workloads
  - Automated rollback
- Make the deployment process observable
  - Real time feedback
  - Alerting
- Write as little YAML as possible
- Manage the whole process from Git

Flagger - The Progressive Delivery Operator

Flagger is a progressive delivery tool that **automates the release** process for applications running on Kubernetes.

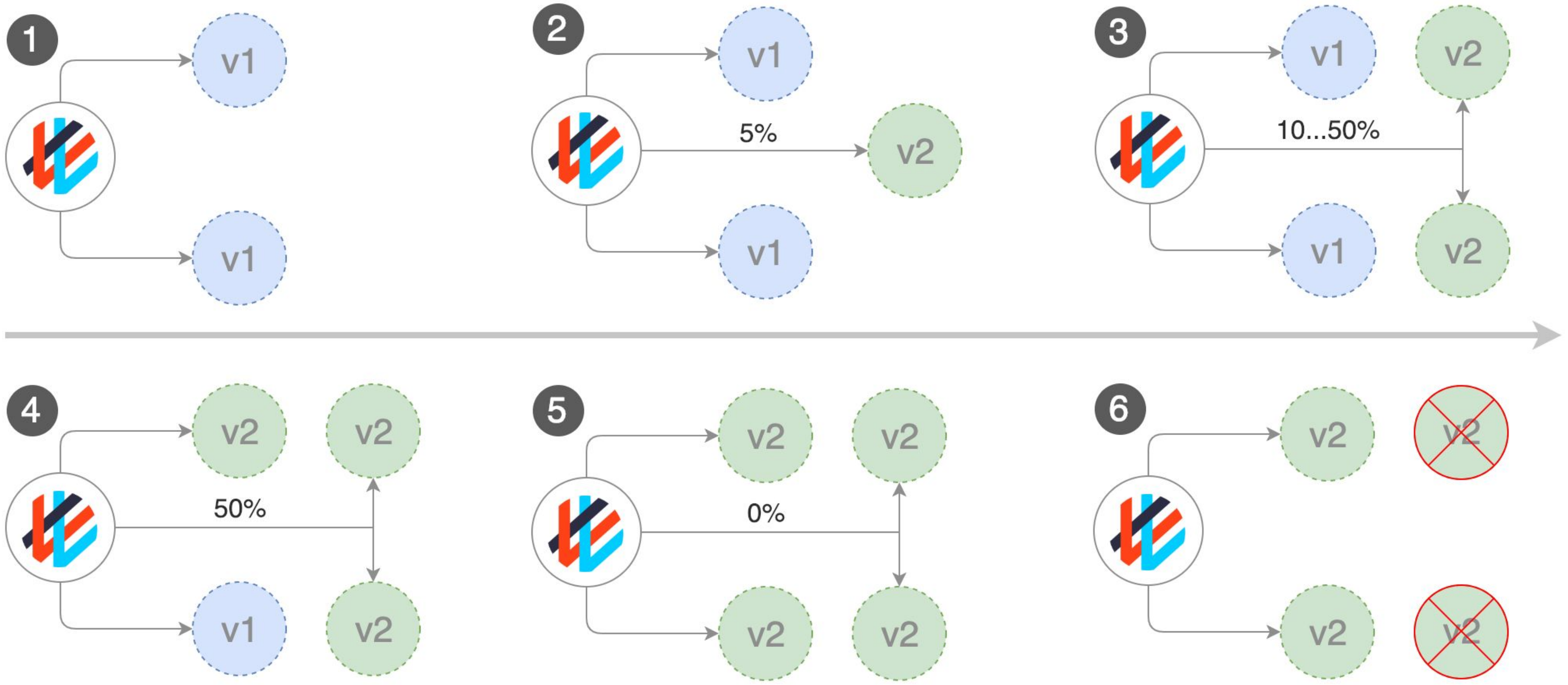
It reduces the risk of introducing a new software version in production by **gradually shifting traffic** to the new version while measuring metrics and running conformance tests.

Flagger comes with a **declarative model** for decoupling the deployment of apps on Kubernetes from the release process.

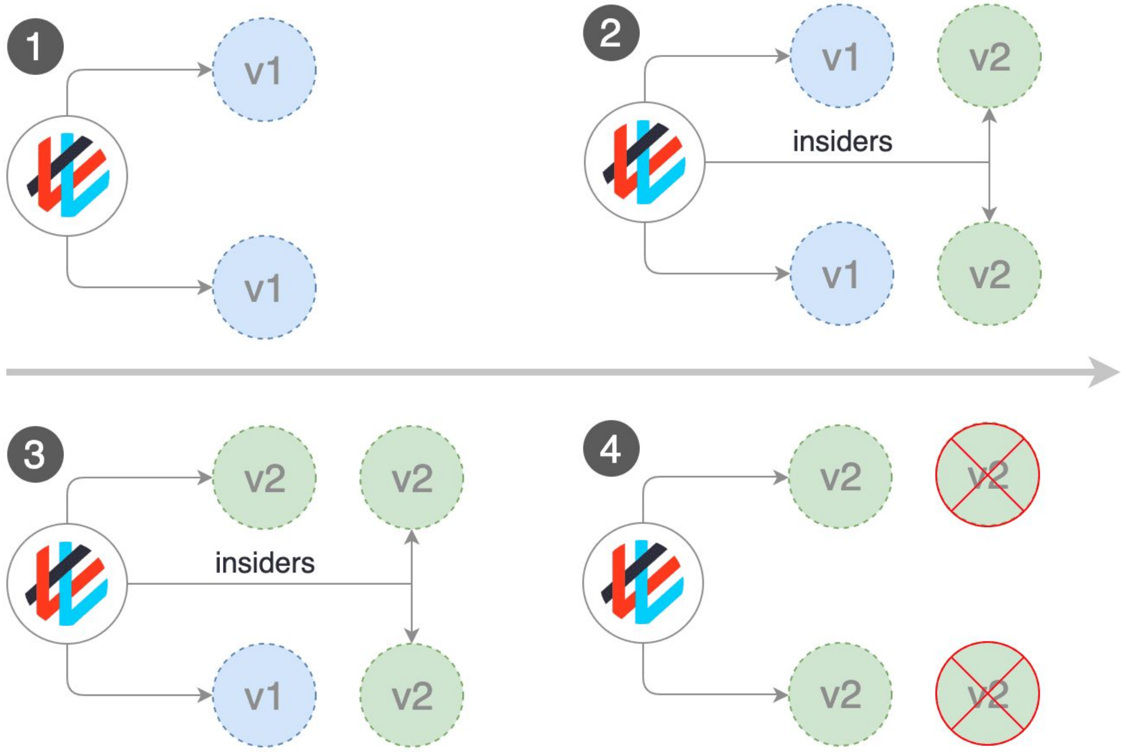
Flagger - Deployment Strategies

- Canary Release (progressive traffic shifting)
  - Applications that expose HTTP or gRPC APIs
- A/B Testing (HTTP headers and cookies traffic routing)
  - User-facing applications that need session affinity
- Blue/Green (traffic mirroring)
  - Idempotent APIs
- Blue/Green (traffic switch)
  - Stateful applications
  - Legacy applications

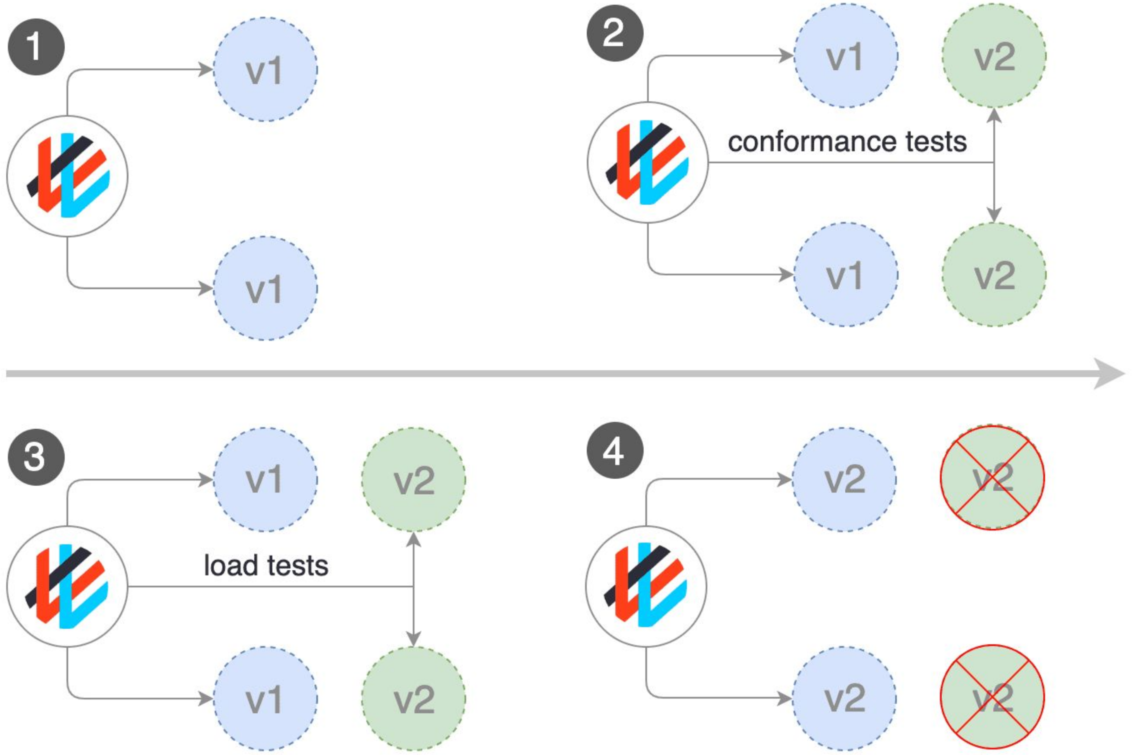
Canary - Deployment Strategy



A/B Testing - Deployment Strategy



Blue/Green - Deployment Strategy



Flagger - Declarative Releases

Specification

- Target Deployment
- Target HPA
- Service
  - Ports
  - Retries
- Analysis
  - Metrics
  - Alerts
  - Webhooks
  - Headers matching

yaml

```yaml
apiVersion: flagger.app/v1beta1
kind: Canary
metadata:
  name: podinfo
spec:
  targetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: podinfo
  service:
    port: 9898
  analysis:
    interval: 5
    threshold: 1
    metrics:
    - name: request-success-rate
      thresholdRange:
        min: 99
      interval: 1m
```

Flagger - Release automation

Manual canary setup

Kubernetes objects

1. Canary Deployment
2. Canary ClusterIP Service
3. Canary Horizontal Pod Autoscaler
4. Primary Deployment
5. Primary ClusterIP Service
6. Primary Horizontal Pod Autoscaler

Service Mesh objects

1. Virtual services
2. HTTP routes
3. Traffic policies
4. Port mappings

Automated canary setup

Kubernetes objects

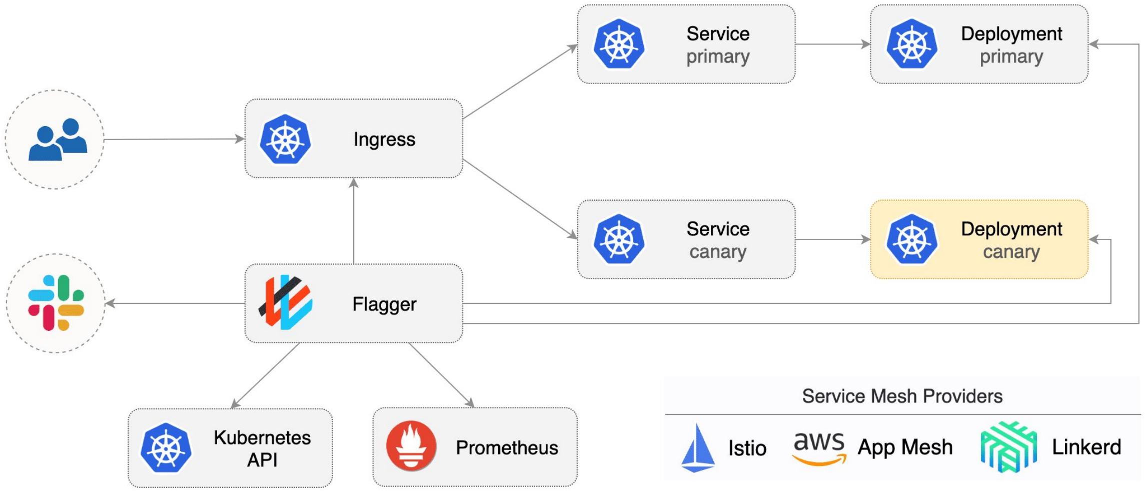
1. Deployment
2. Horizontal Pod Autoscaler

Flagger objects

1. Canary

Flagger - Traffic management

- Service Mesh
  - Istio
  - Linkerd
  - App Mesh
- Ingress Controllers
  - Contour
  - Gloo
  - NGINX



Flagger - Validation process

Flagger lets you define **key performance indicators** and **thresholds**. The decision to pause the traffic shift, abort or promote a canary is based on:

- Deployment health status
- Request success rate percentage (built-in metric)
- Request latency average value (built-in metric)
- Custom checks (metric templates)
- Webhooks (integration testing, load testing, etc)

Flagger - Metric Templates

Metrics Providers

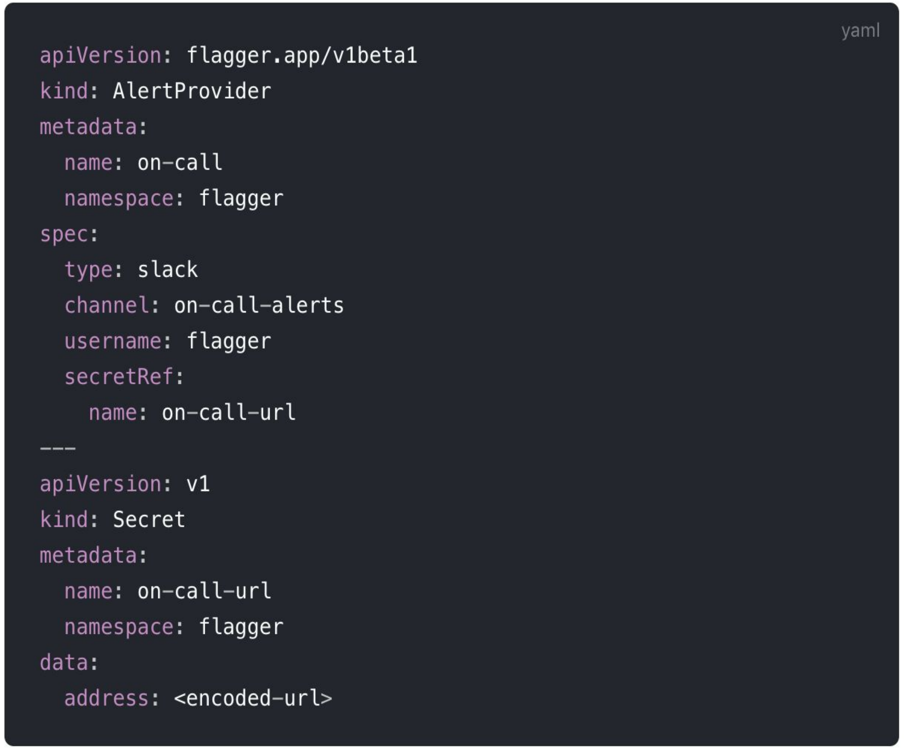
- Prometheus
- Datadog
- CloudWatch

```yaml
apiVersion: flagger.app/v1beta1
kind: MetricTemplate
metadata:
  name: latency
  namespace: istio-system
spec:
  provider:
    type: prometheus
    address: "http://prometheus.istio-system:9090"
  query: |
    histogram_quantile(
        0.95,
        sum(
            rate(
                istio_request_duration_milliseconds_bucket{
                    reporter="destination",
                    destination_workload_namespace="{{ namespace }}",
                    destination_workload=~"{{ target }}"
                }[{{ interval }}]
            )
        ) by (le)
    )
```

Flagger - Alerting

Alert Providers

- Slack
- Microsoft Teams
- Discord
- Rocket Chat



yaml

```yaml
apiVersion: flagger.app/v1beta1
kind: AlertProvider
metadata:
  name: on-call
  namespace: flagger
spec:
  type: slack
  channel: on-call-alerts
  username: flagger
  secretRef:
    name: on-call-url
---
apiVersion: v1
kind: Secret
metadata:
  name: on-call-url
  namespace: flagger
data:
  address: <encoded-url>
```

Flagger - Testing Webhooks

Test runner service

- Load testing
  - Hey (HTTP)
  - WRK (HTTP)
  - GHZ (gPRC)
- Conformance testing
  - Helm test
  - Bash Bats
  - Bring your own

yaml

```yaml
webhooks:
  - name: "helm test"
    type: pre-rollout
    url: http://flagger-helmtester.flagger/
    timeout: 3m
    metadata:
      type: "helmv3"
      cmd: "test podinfo -n test"
  - name: "load test"
    type: rollout
    url: http://flagger-loadtester.test/
    timeout: 15s
    metadata:
      cmd: "hey -z 1m -q 5 -c 2 http://podinfo-canary.test:9898/"
```

Flagger - Manual Gating

Gating service

- Confirm rollout
- Confirm promotion
- Confirm rollback

yaml

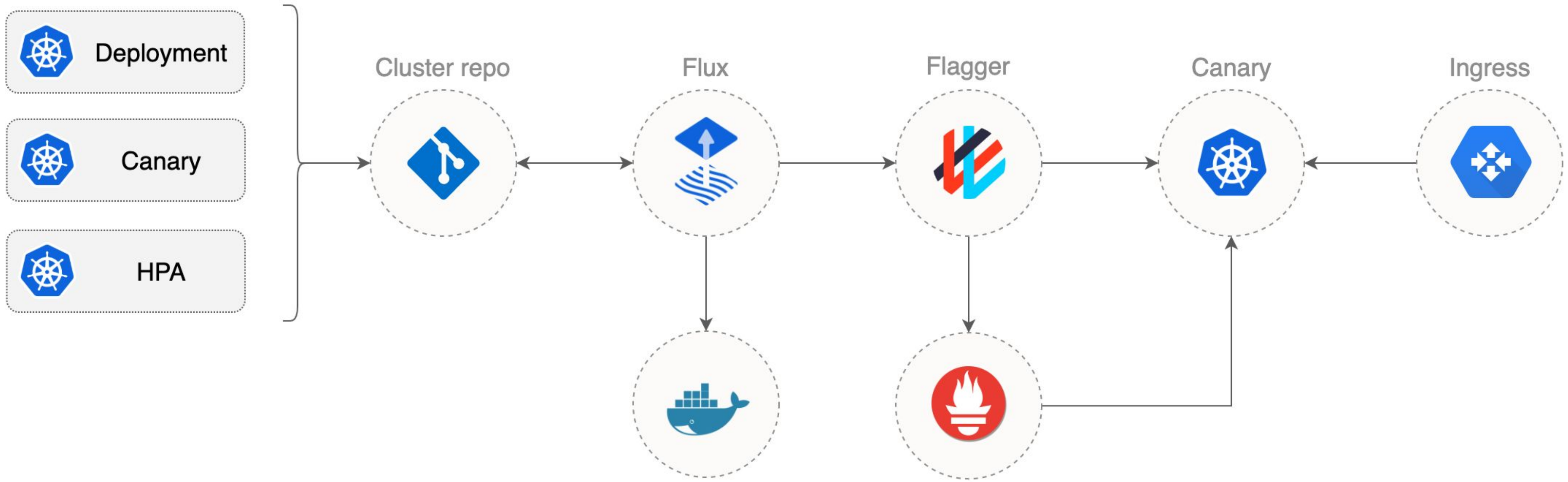
```yaml
webhooks:
  - name: "start gate"
    type: confirm-rollout
    url: http://flagger-gatekeeper.test/gate/check
  - name: "promotion gate"
    type: confirm-promotion
    url: http://flagger-gatekeeper.test/gate/check
  - name: "rollback gate"
    type: rollback
    url: http://flagger-gatekeeper.test/rollback/check
```

sh

```sh
curl -d '{"name": "podinfo","namespace":"test"}' \
 http://flagger-gatekeeper/gate/open

curl -d '{"name": "podinfo","namespace":"test"}' \
 http://flagger-gatekeeper/gate/close
```

Flux & Flagger - GitOps pipeline



Flagger - Roadmap

- Conformance testing with Kubernetes Job
- Manual gating dedicated service
- add more metric providers like Stackdriver and InfluxDB
- extend support for other service meshes that implement SMI
- add support for Kubernetes Ingress v2

Hands-on Workshops

AWS App Mesh on EKS
https://eks.handson.flagger.dev

Linkerd and NGINX ingress
https://helm.workshop.flagger.dev

Istio 1.5
https://github.com/stefanprodan/gitops-istio

Links

Flagger Repo
https://github.com/weaveworks/flagger

Flagger Docs
https://docs.flagger.app