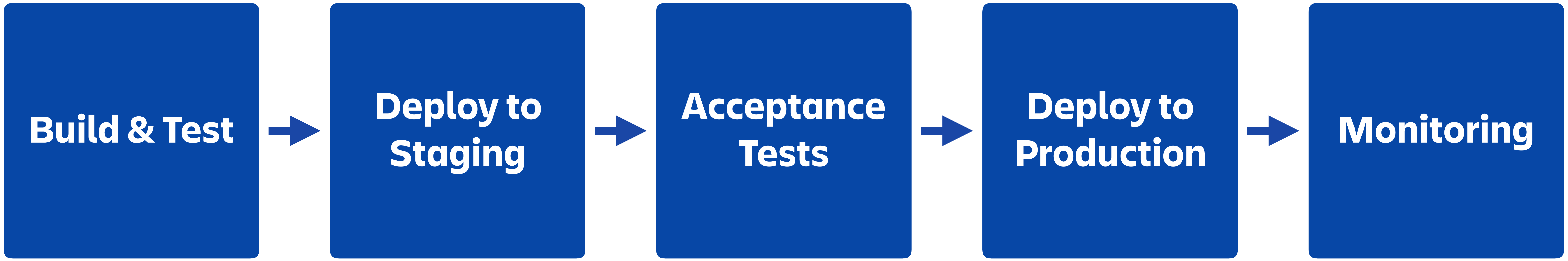
Custom Deployment Strategies for Kubernetes

NAIL ISLAMOV | SENIOR DEVELOPER | @NILEBOX

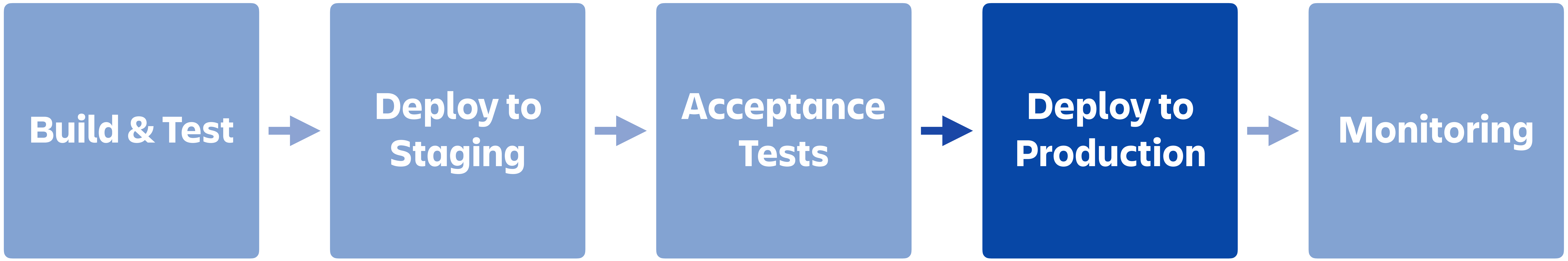
CONTINUOUS DELIVERY

Continuous delivery is an approach where teams release products frequently and predictably from source code repository to production in an automated fashion.

CI/CD PIPELINE



CI/CD PIPELINE



Deployment

Deployment strategies

Recreate

Kill all existing pods before creating new ones.

RollingUpdate

Gradually scale down the old ReplicaSets and scale up the new one.

Deployment strategies

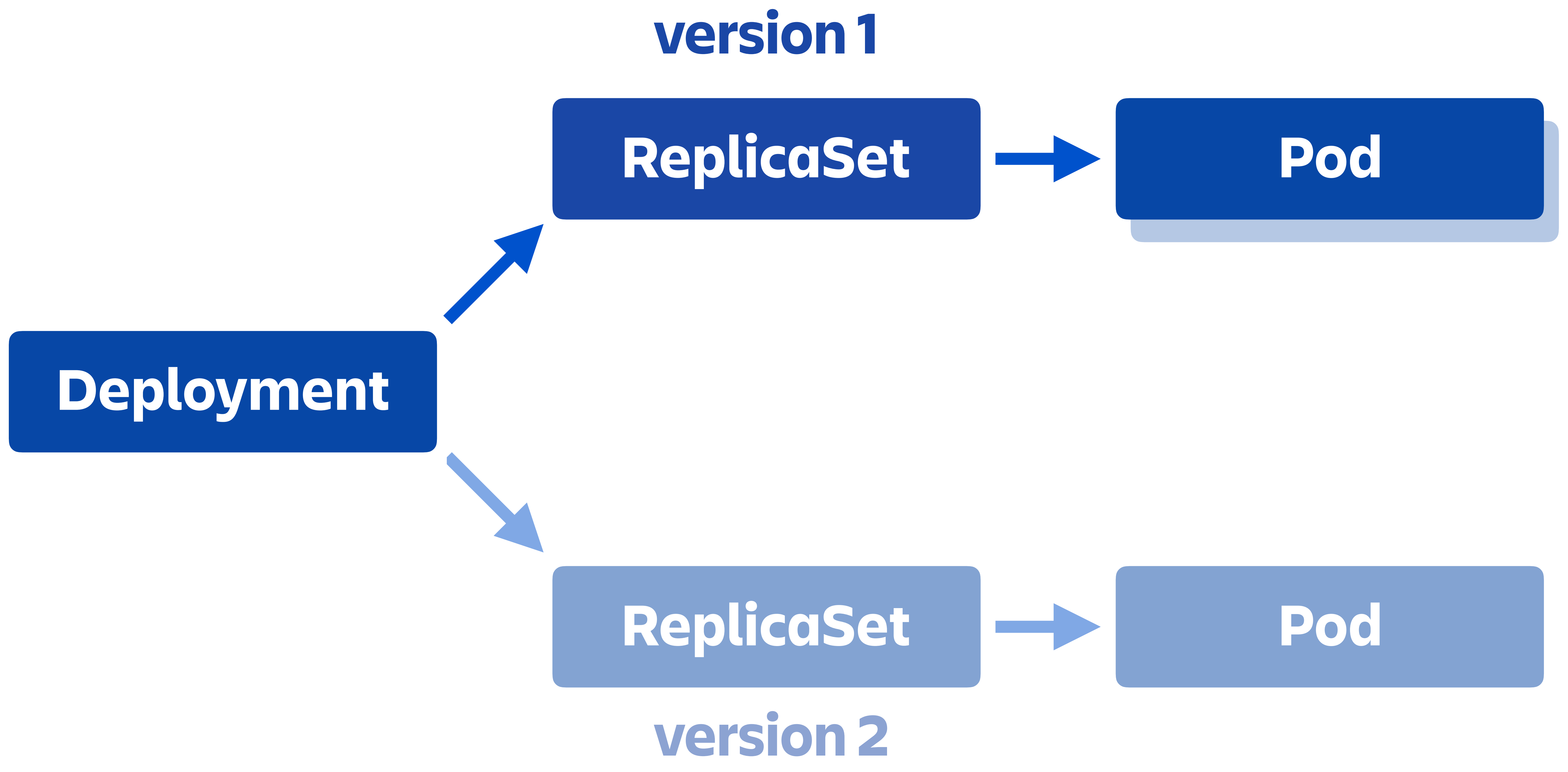
Recreate

Kill all existing pods before creating new ones.

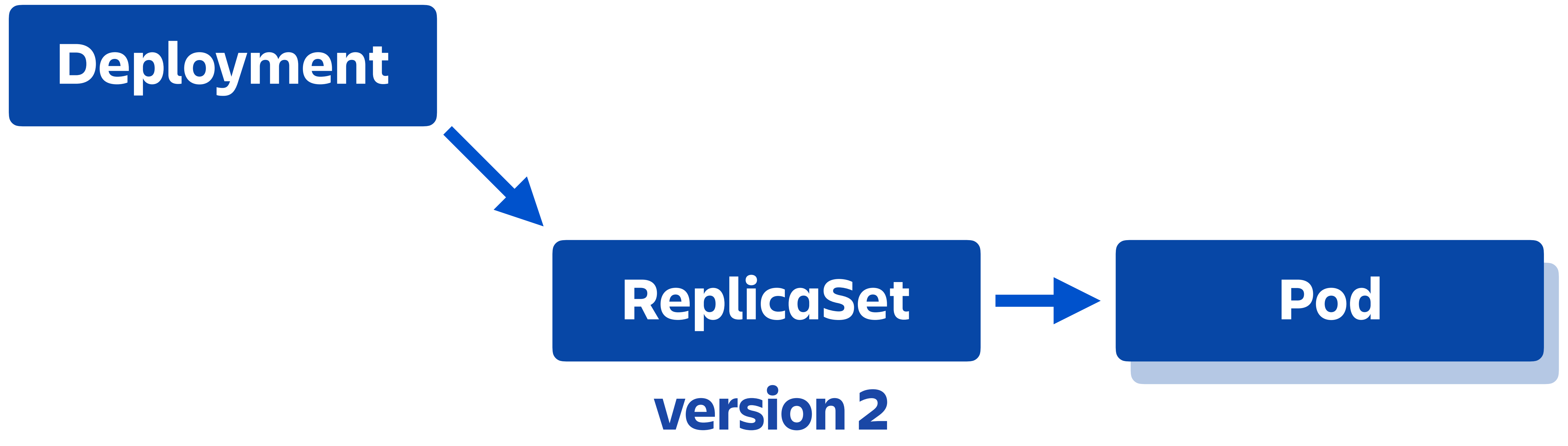
RollingUpdate

Gradually scale down the old ReplicaSets and scale up the new one.

ROLLING UPDATE



ROLLING UPDATE



Continuous Deployment

ROLLING UPDATE: TRAFFIC TIMELINE

v1
v2
v1
v2
v1
v2
v1
v2
v1
v2
v1
v2
v1
v2
v1
v2
v1
v2
v1
v2
v1
v2

ROLLING UPDATE: ROLLBACK

v1
v2
v1
v2
v1
v2
v1
v2
v2
v2
v2
v1
v2
v1
v2
v1
v1

How do we detect issues in production?

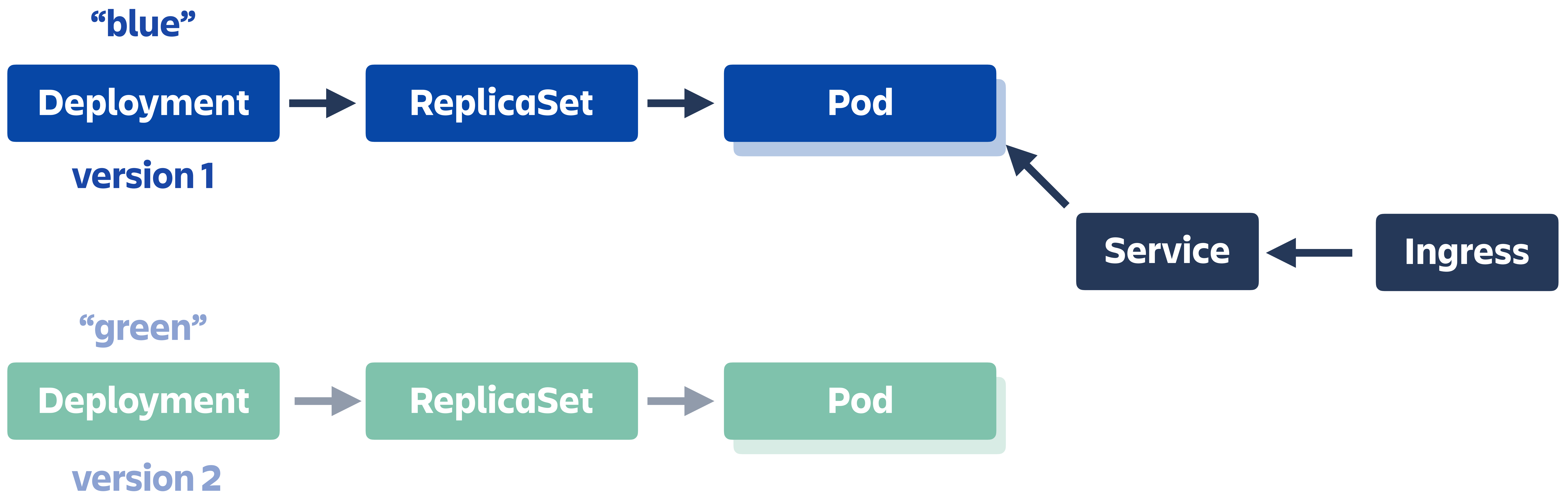
Metrics.

How do we reduce impact in case of a bad release?

Custom deployment strategies.

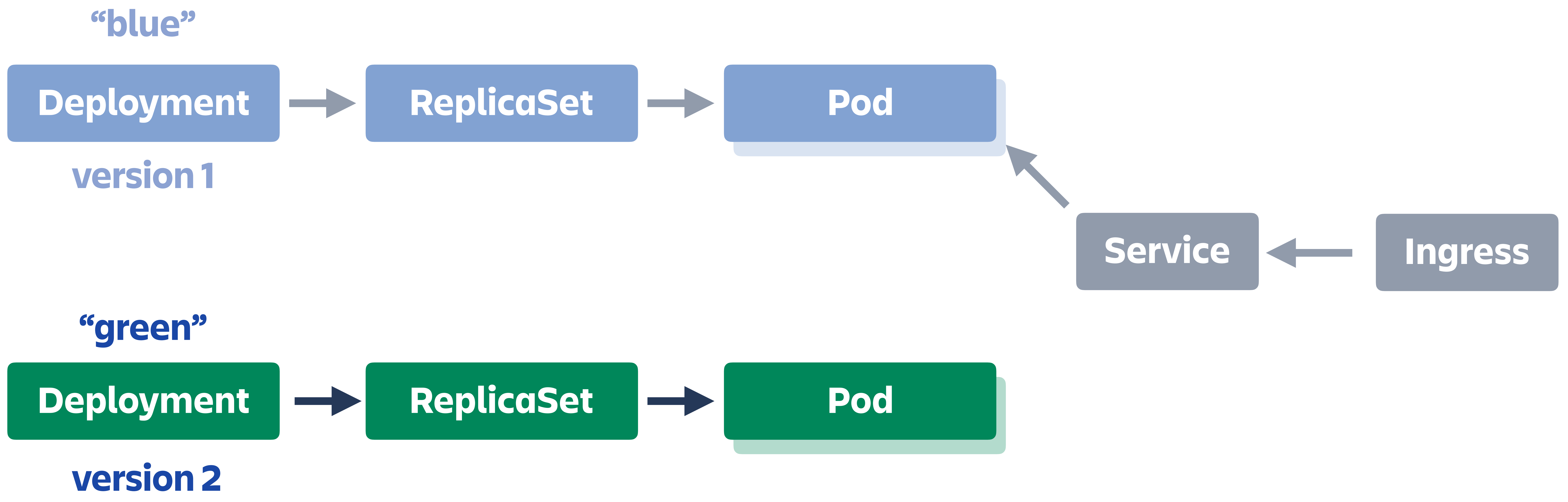
Custom Deployment Strategies

BLUE-GREEN DEPLOYMENT



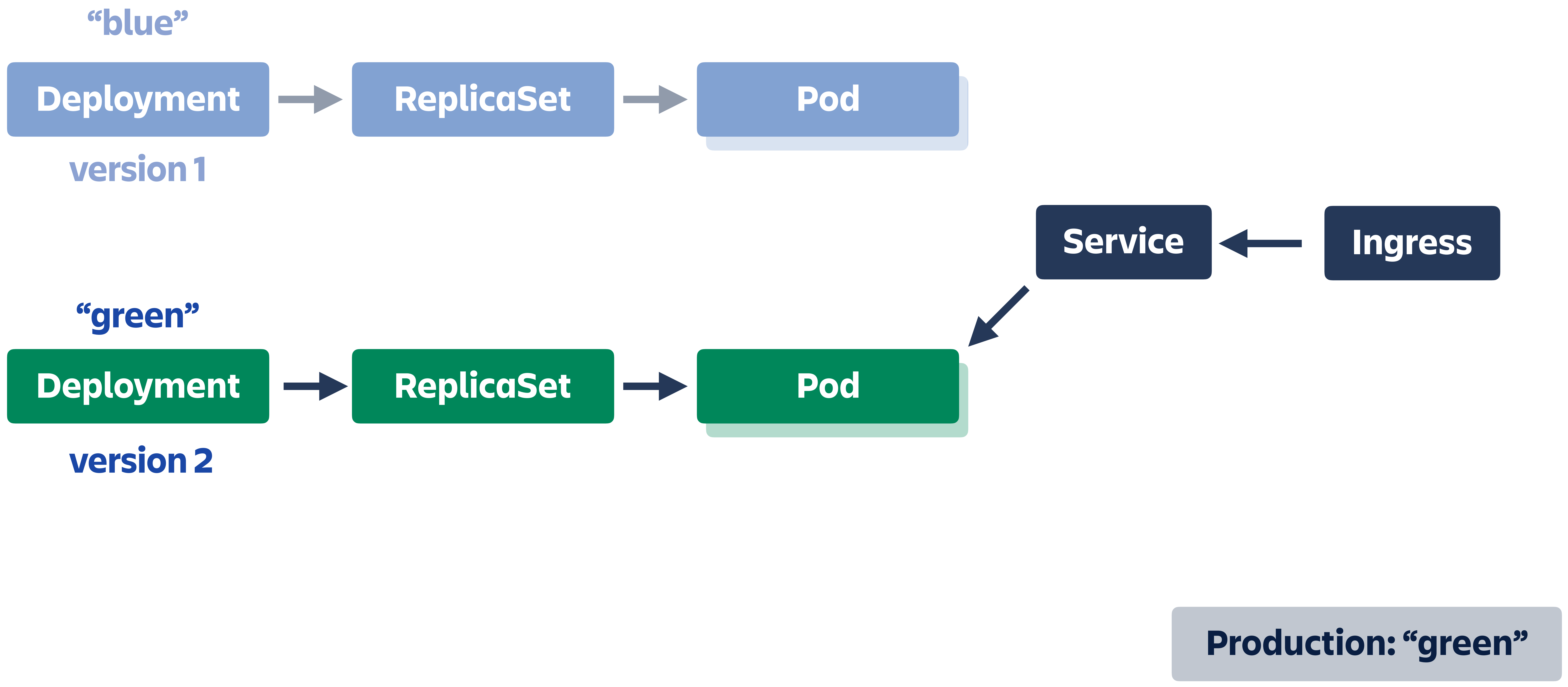
Production: “blue"

BLUE-GREEN DEPLOYMENT



Production: "blue"

BLUE-GREEN DEPLOYMENT



BLUE-GREEN DEPLOYMENT: TRAFFIC TIMELINE

v1
v1
v1
v1
v2
v2
v2
v2
v2
v2
v3

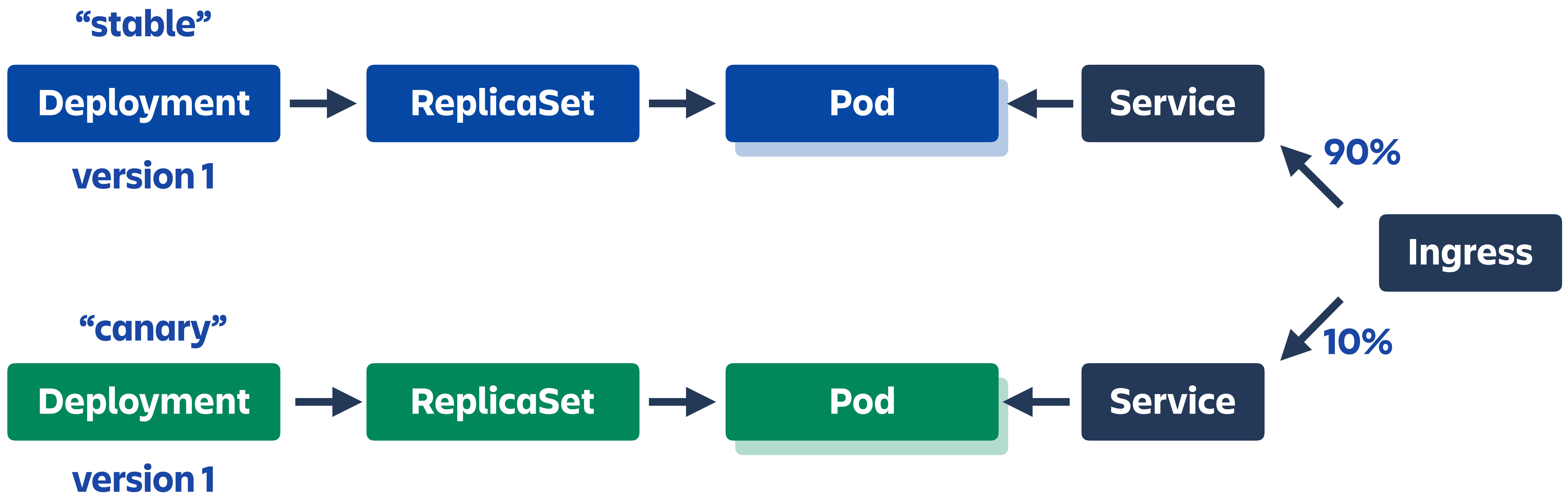
BLUE-GREEN DEPLOYMENT: ROLLBACK

v1
v1
v1
v1
v2
v2
v1
v1
v1
v1
v1

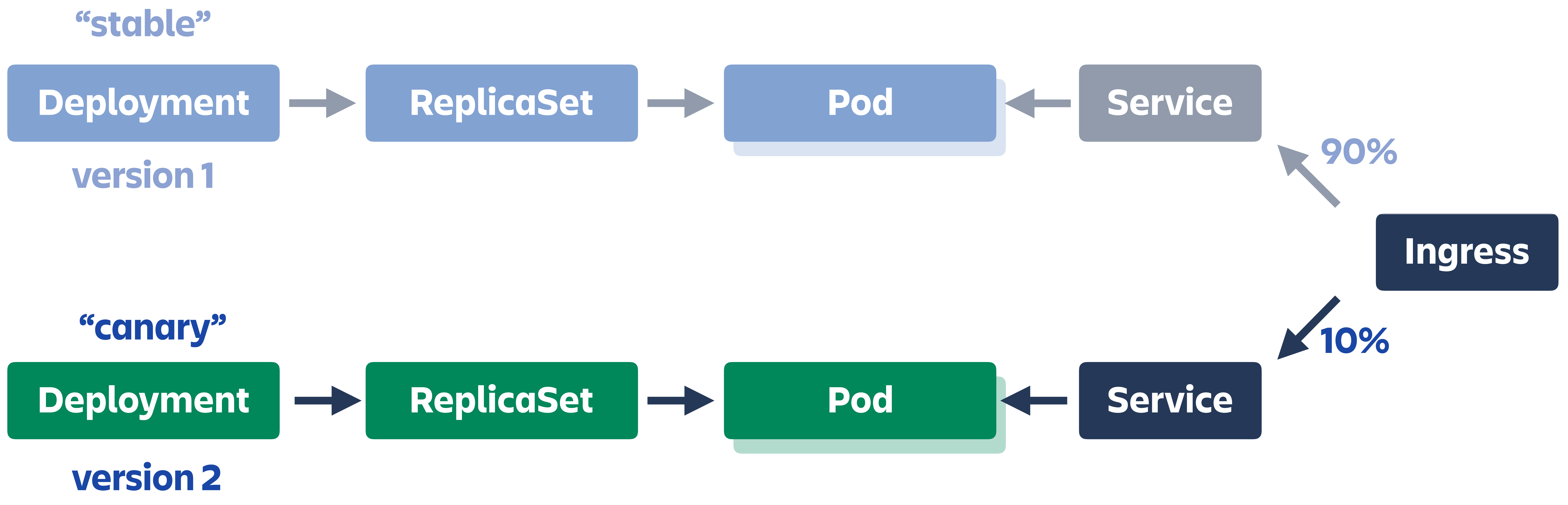
Canary release is a technique to reduce the risk of introducing a new software version in production by slowly rolling out the change to a small subset of users before rolling it out to the entire infrastructure and making it available to everybody.

MARTINFOWLER.COM

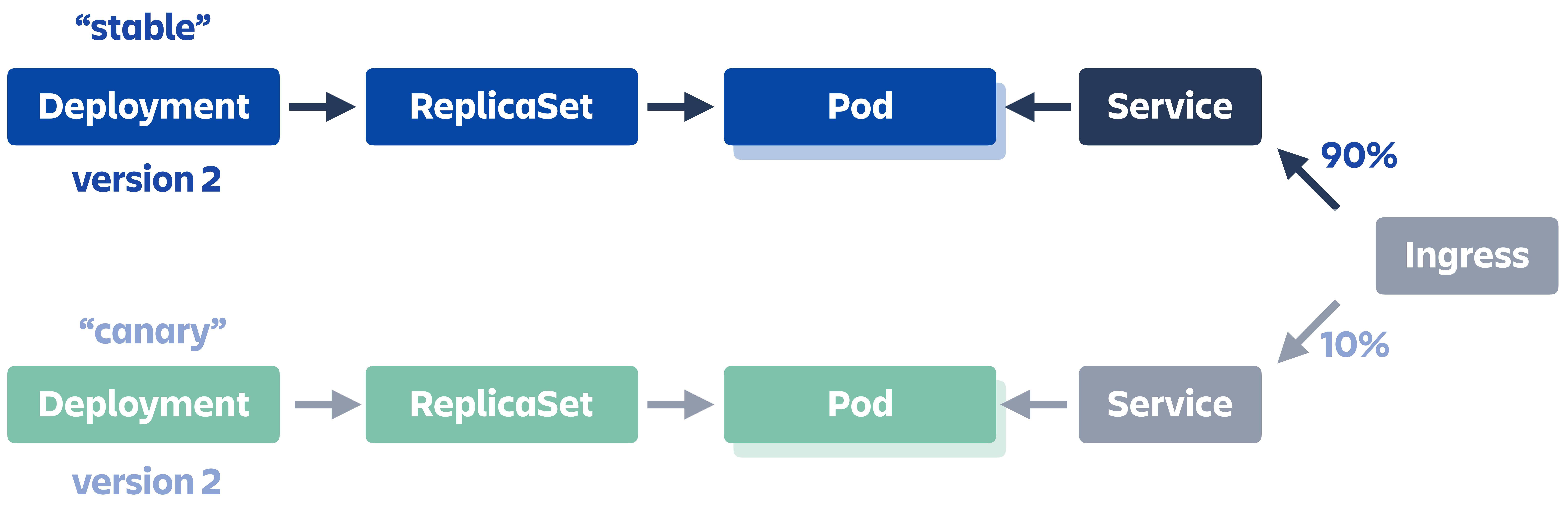
CANARY DEPLOYMENT



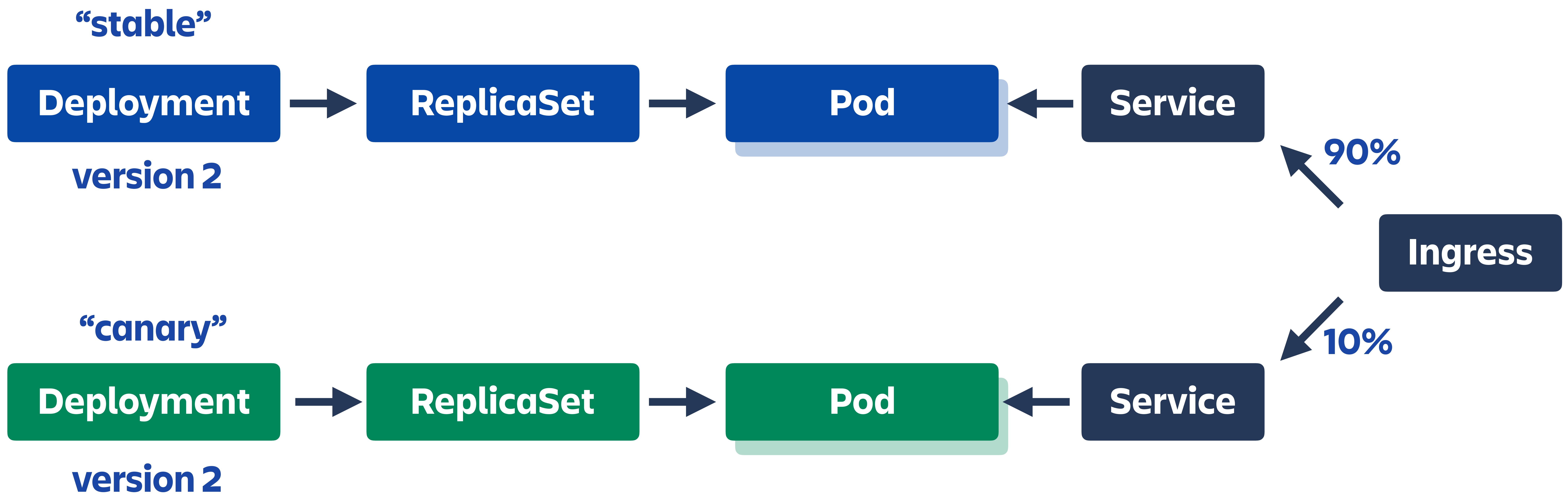
CANARY DEPLOYMENT



CANARY DEPLOYMENT



CANARY DEPLOYMENT



CANARY DEPLOYMENT: TRAFFIC TIMELINE

v1
v2
v1
v2
v1
v2
v1
v2
v1
v2
v1
v2
v1
v2
v1
v2
v1
v2
v1
v2

CANARY DEPLOYMENT: ROLLBACK

v1
v2
v1
v2
v1
v2
v1
v2
v1
v1
v1
v1
v1
v1
v1

Canary Deployment Controller

How do we automate the deployment rollout?

Scripts in CI/CD tool.

EXAMPLE

```
kubectl apply -f deployment-canary.yaml
kubectl apply -f deployment-stable.yaml
kubectl apply -f service-canary.yaml
kubectl apply -f service-stable.yaml
kubectl apply -f ingress.yaml
```

EXAMPLE

```
kubectl apply -f deployment-canary.yaml
sleep 5m # wait for rollout to finish
# check if application is healthy
curl http://metrics:9090/my-metric
# proceed or rollback
if ...

kubectl apply -f deployment-stable.yaml
```

Can we do better?

How do we automate the deployment rollout?

~~Scripts in CI/CD tool.~~
CRD controller?

Benefits of CRDs

Declarative

Describes the desired state, not the steps to reach it

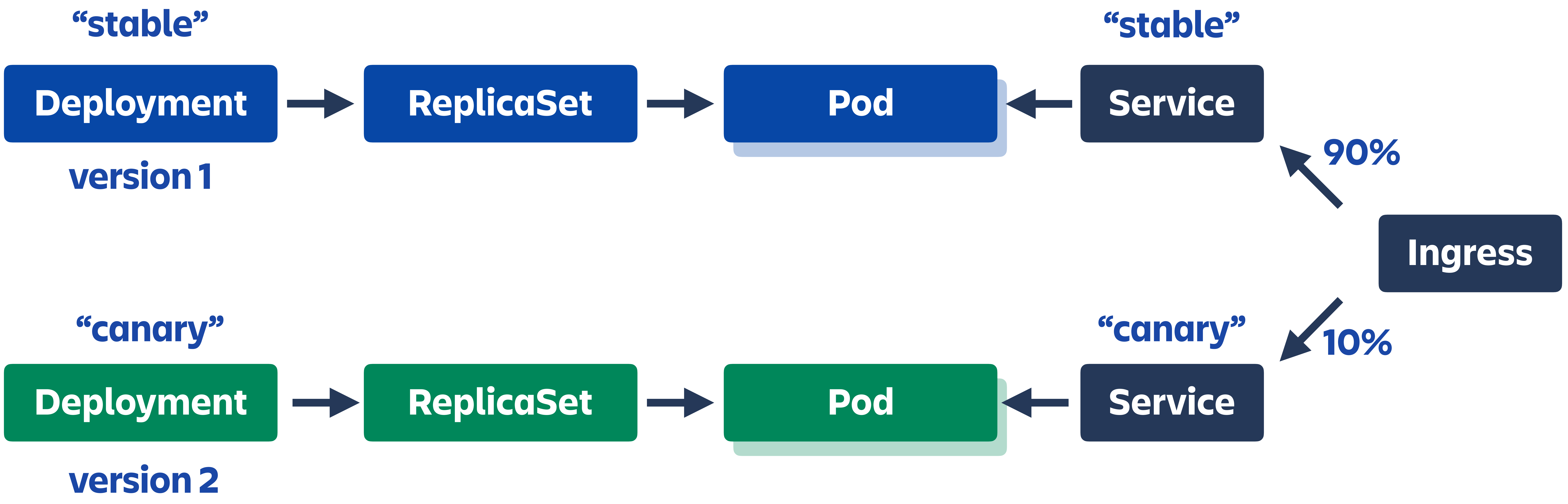
Self-healing

Reconciliation loop will keep retrying until reaching the final state

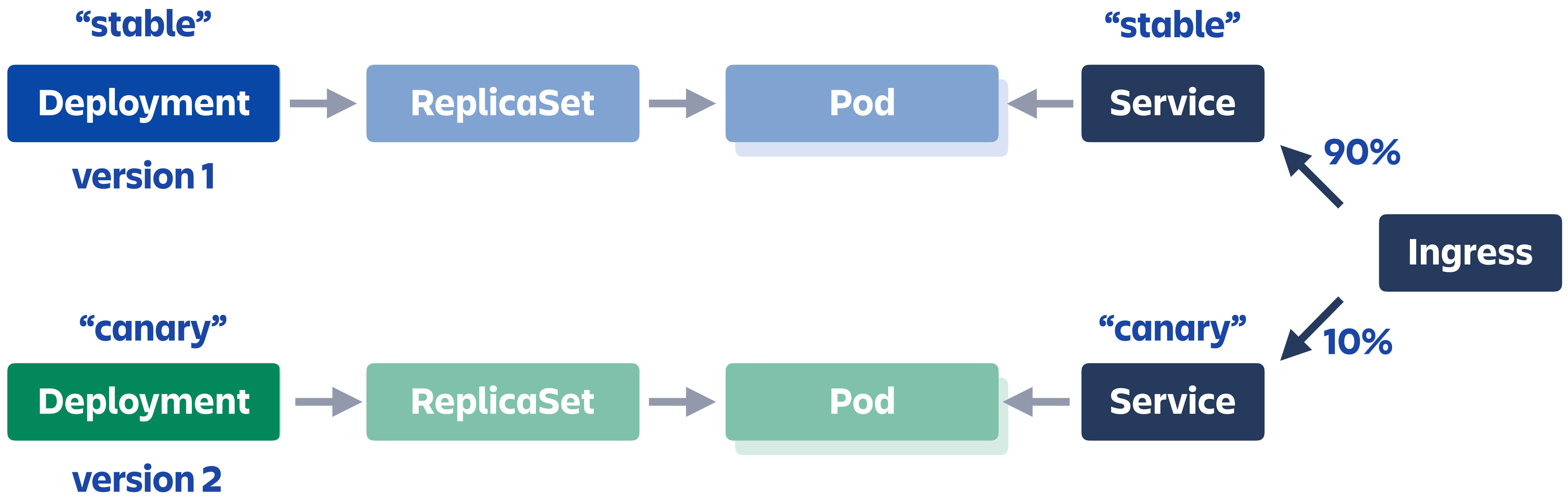
Reusable

Building block that can be used together with other Kubernetes resources

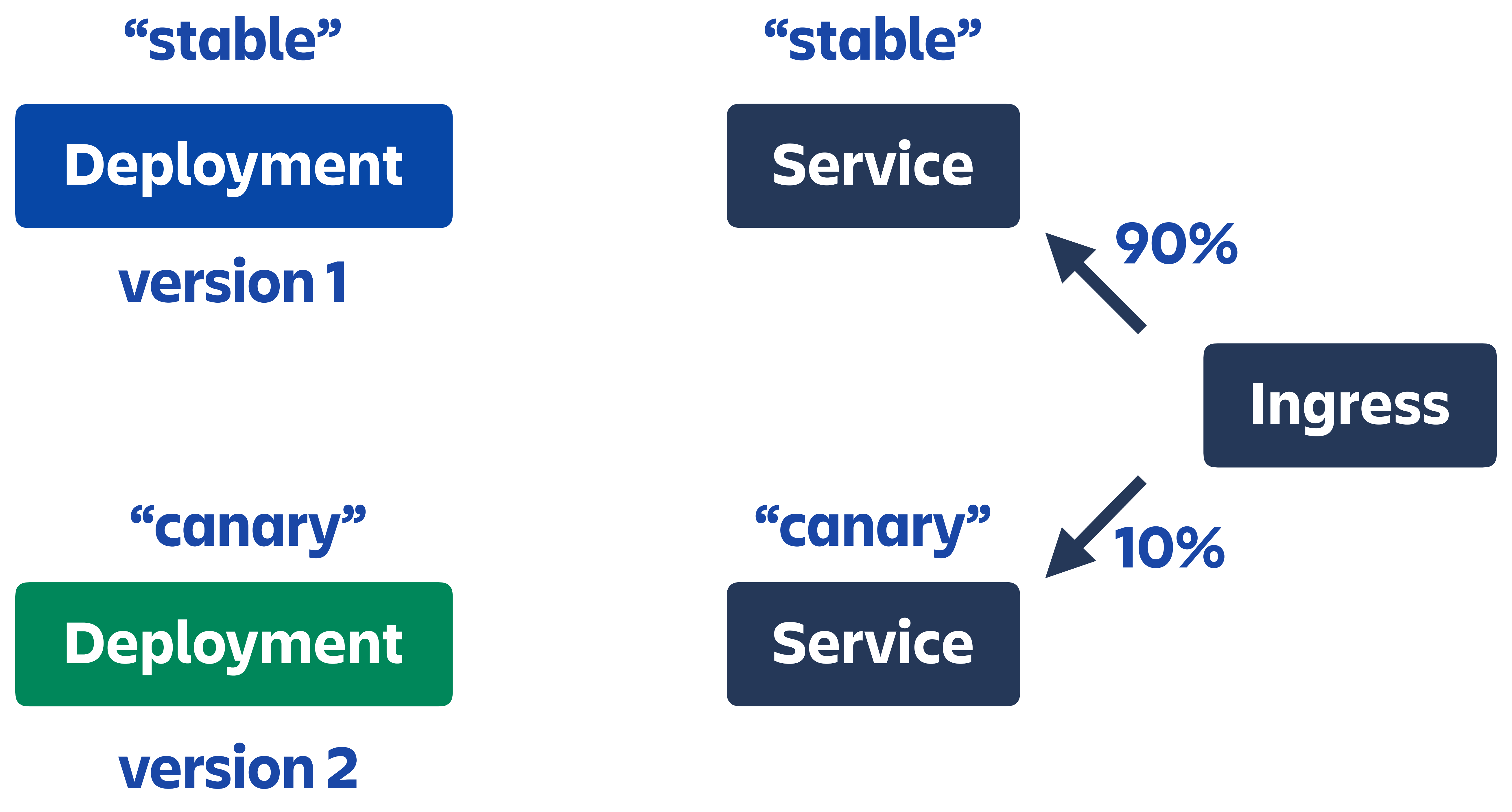
CANARY DEPLOYMENT



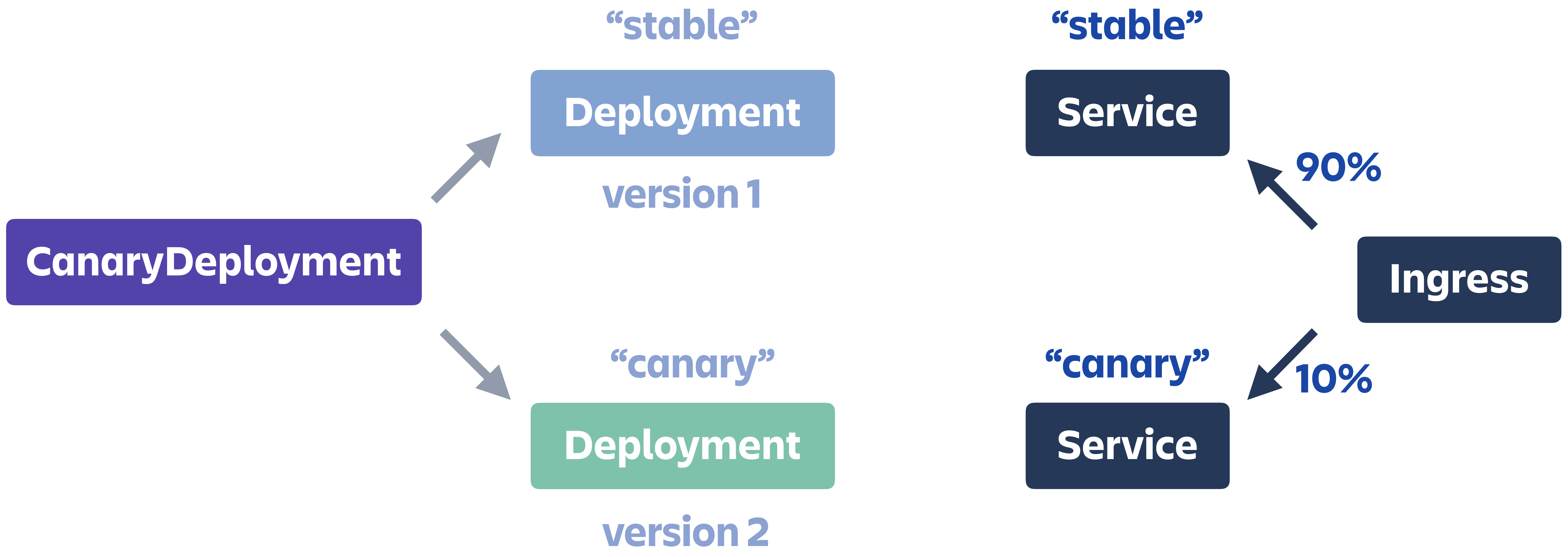
CANARY DEPLOYMENT



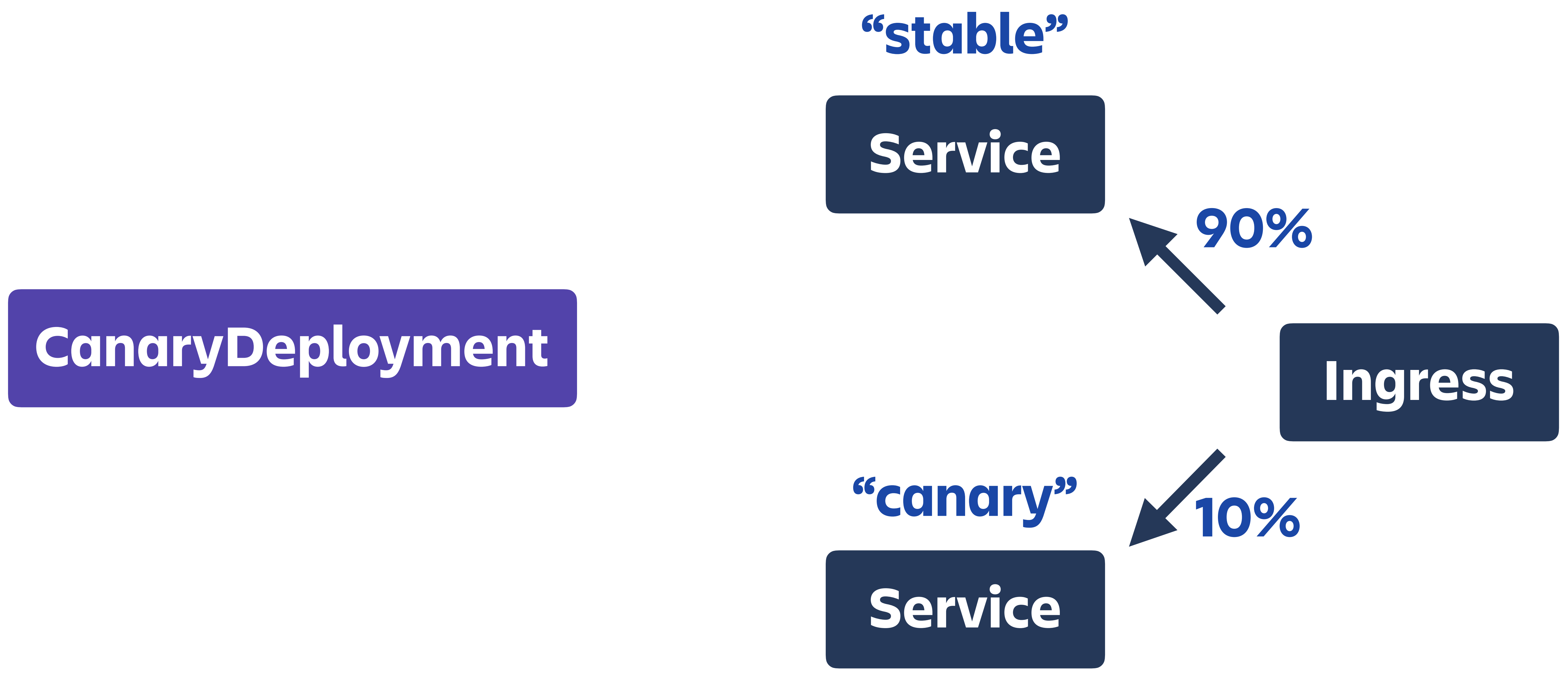
CANARY DEPLOYMENT



CANARY DEPLOYMENT



CANARY DEPLOYMENT



DECLARATIVE CONTINUOUS DEPLOYMENT

```
kubectl apply -f canarydeployment.yaml
kubectl apply -f service-canary.yaml
kubectl apply -f service-stable.yaml
kubectl apply -f ingress.yaml
```

How will CanaryDeployment controller detect a bad release?

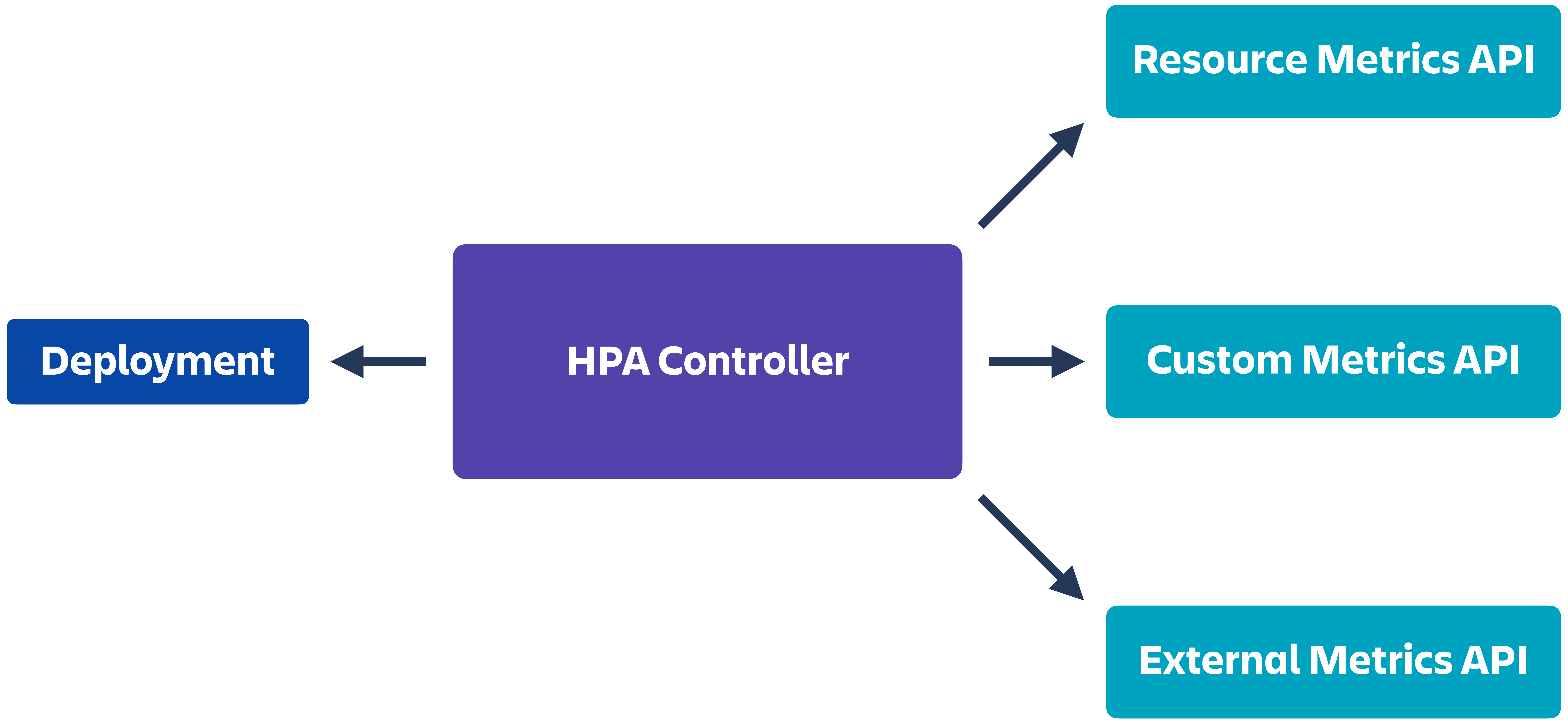
Metrics.

Kubernetes Metrics APIs

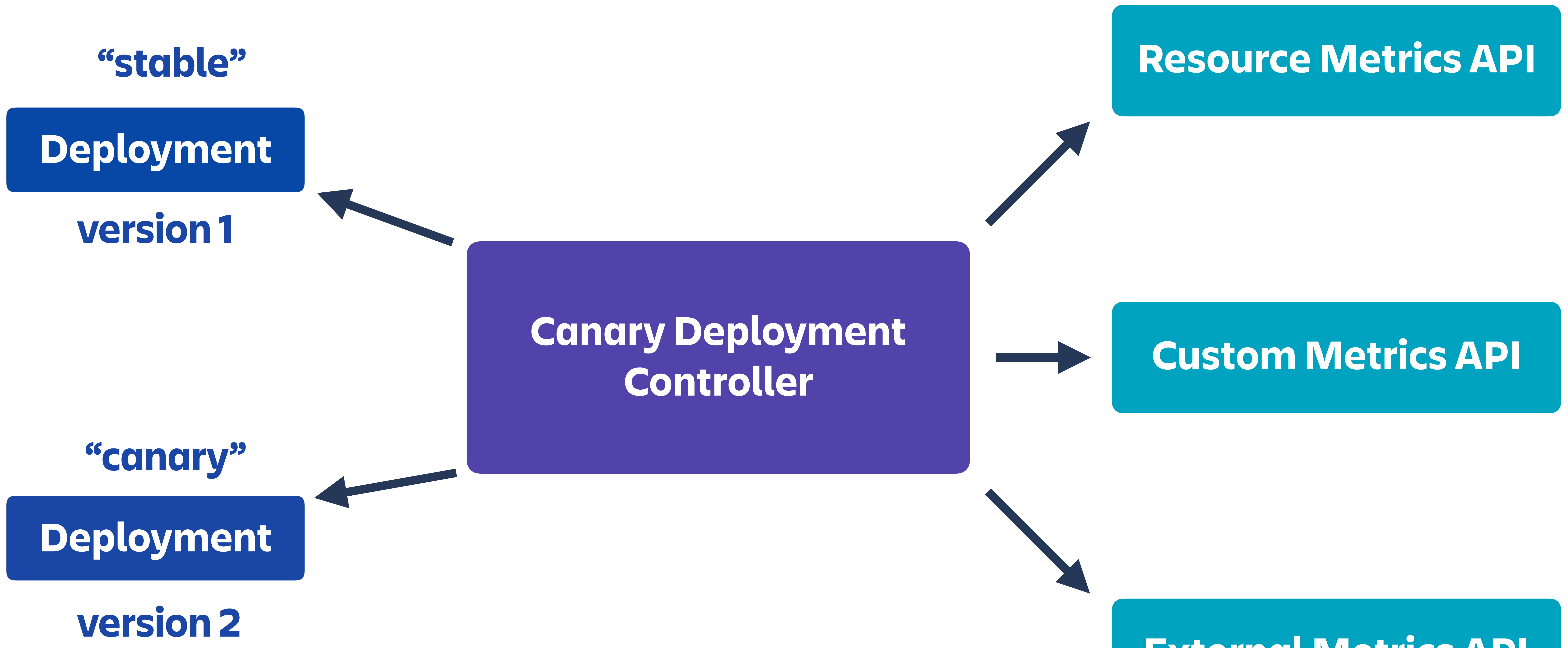
HORIZONTAL POD AUTOSCALER (HPA)

Horizontal Pod Autoscaler is the primary consumer of Kubernetes Metrics APIs at the moment.

HPA CONTROLLER



CANARY DEPLOYMENT CONTROLLER



RESOURCE METRICS API

Metrics for Pods and Nodes

- CPU
- Memory

CUSTOM METRICS API

Arbitrary metrics for any Kubernetes resource

- Pod
- Service
- Ingress

CUSTOM METRICS API ADAPTERS

Prometheus Adapter
https://github.com/DirectXMan12/k8s-prometheus-adapter

Stackdriver (GCP)
https://cloud.google.com/monitoring/custom-metrics/

Azure Kubernetes Metrics Adapter
https://github.com/Azure/azure-k8s-metrics-adapter

Datadog Cluster Agent
https://github.com/DataDog/datadog-agent/blob/master/docs/cluster-agent/CUSTOM_METRICS_SERVER.md

Custom Metrics Adapter Server Boilerplate
https://github.com/kubernetes-incubator/custom-metrics-apiserver

EXTERNAL METRICS API

Arbitrary metrics from outside of Kubernetes cluster

- Amazon SQS queue size (CloudWatch)
- Google Cloud Pub/Sub undelivered messages (Stackdriver)

CanaryDeployment CRD

DEPLOYMENT

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: foo
spec:
  replicas: 5
  selector: ... # Pod selector
  template: ... # Pod template
```

CANARY DEPLOYMENT

```yaml
apiVersion: kanarini.nilebox.github.com/v1alpha1
kind: CanaryDeployment
metadata:
  name: foo
spec:
  selector: ... # Pod selector
  template: ... # Pod template
  tracks:
    canary: ... # "canary" track settings
    stable: ... # "stable" track settings
```

CANARY DEPLOYMENT

```yaml
apiVersion: kanarini.nilebox.github.com/v1alpha1
kind: CanaryDeployment
metadata:
  name: foo
spec:
  selector: ... # Pod selector
  template: ... # Pod template
  tracks:
    canary: ... # "canary" track settings
    stable: ... # "stable" track settings
```

CANARY DEPLOYMENT

```yaml
apiVersion: kanarini.nilebox.github.com/v1alpha1
kind: CanaryDeployment
metadata:
  name: foo
spec:
  selector: ... # Pod selector
  template: ... # Pod template
  tracks:
    canary: ... # "canary" track settings
    stable: ... # "stable" track settings
```

CANARY DEPLOYMENT

```
tracks:
  canary:
    replicas: 1
    labels:
      track: canary
    metricsCheckDelaySeconds: 120
    metrics: ... # List of metrics to check against
  stable:
    replicas: 5
    labels:
      track: stable
```

CANARY DEPLOYMENT

```yaml
metricsCheckDelaySeconds: 120
metrics:
- type: Object
  object:
    describedObject:
      kind: Service
      name: "foo-canary"
    metric:
      name: "request_failure_rate:1m"
    target:
      type: Value
      value: 0.1
```

CANARY DEPLOYMENT

```yaml
metricsCheckDelaySeconds: 120
metrics:
- type: Object
  object:
    describedObject:
      kind: Service
      name: "foo-canary"
    metric:
      name: "request_failure_rate:1m"
    target:
      type: Value
      value: 0.1
```

CANARY DEPLOYMENT

```
metricsCheckDelaySeconds: 120
metrics:
- type: Object
  object:
    describedObject:
      kind: Service
      name: "foo-canary"
    metric:
      name: "request_failure_rate:1m"
    target:
      type: Value
      value: 0.1
```

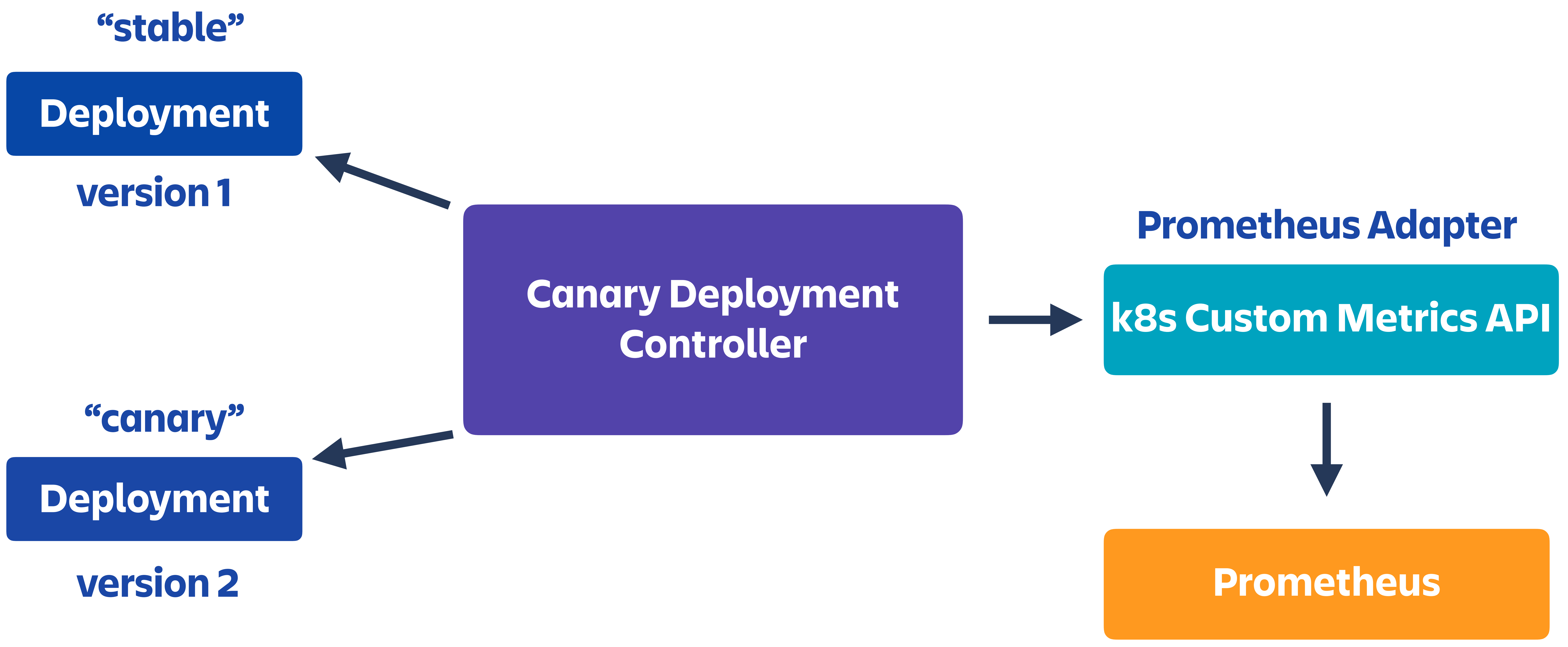
CANARY DEPLOYMENT

```yaml
metricsCheckDelaySeconds: 120
metrics:
- type: Object
  object:
    describedObject:
      kind: Service
      name: "foo-canary"
    metric:
      name: "request_failure_rate:1m"
    target:
      type: Value
      value: 0.1
```

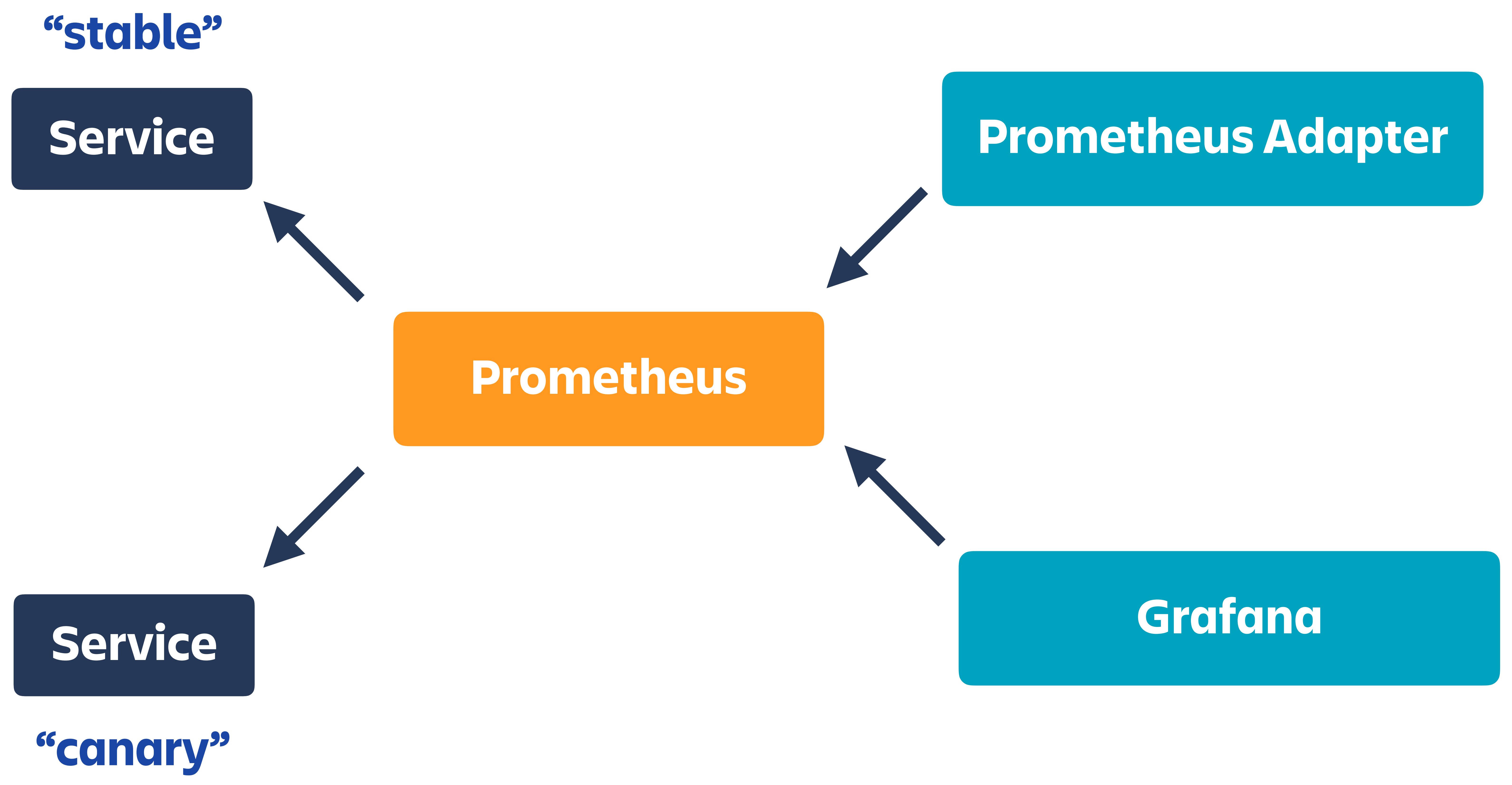
Demo

Kanarini CRD Controller

EXAMPLE


"stable"
Deployment
version 1
"canary"
Deployment
version 2
Canary Deployment Controller
Prometheus Adapter
k8s Custom Metrics API
Prometheus

EXAMPLE


"stable"
Service
Prometheus Adapter
Prometheus
Service
Grafana
"canary"

Demo script is available at
https://github.com/nilebox/kanarini

LINKS

Kanarini (CanaryDeployment CRD Controller)
https://github.com/nilebox/kanarini

Prometheus Adapter for Custom Metrics API
https://github.com/DirectXMan12/k8s-prometheus-adapter

Prometheus Operator Quickstart
https://github.com/coreos/prometheus-operator/tree/master/contrib/kube-prometheus

Heptio Contour (Ingress Controller)
https://github.com/heptio/contour

Key takeaways for CRDs

Reuse existing resources
No need to reinvent the wheel.

Keep it simple
Solve a minimal subset of a problem at once.

Use abstractions
Generic APIs are reusable.

Use the power of open source
Read existing code and share your own code.

Thank you!

NAIL ISLAMOV | SENIOR DEVELOPER | @NILEBOX