



# Laser Focused deployments with Argo Rollouts

**Kubecon US 2024**

Kostis Kapelonis | November 2024

# Kostis Kapelonis



## Developer Advocate (Octopus Deploy/Codefresh)

Argo Maintainer (Argo CD, Argo Rollouts)

Co-author GitOps certification

<http://learning.codefresh.io>



# Codefresh GitOps by Octopus Deploy

- **Argo Control plane and Promotions**  
Model app promotions
- **Argo Enterprise Support & TAM**  
Expert help from Argo Maintainers
- **Hardened Security Distro for Argo**  
Aggressive patching SLA from #1 security maintainers
- **#1 GitOps Training & Certification**  
Over 23k students...

The screenshot displays the Codefresh web interface for a deployment pipeline. The left sidebar contains navigation options: Getting Started, Home Dashboard, PIPELINES (Pipelines Dashboard, Projects, Pipelines, Builds, Steps), WORKFLOWS, OPS (GitOps Classic, GitOps Apps, Products, Environments), Releases (highlighted), Helm Boards, Kubernetes Services, Helm Releases, and ARTIFACTS & INSIGHT (DORA Metrics, Images, Helm Charts). The main content area shows a release for 'my-product' (cloud deployment) with ID 'id281ghj3971' and version 'v.1.30.7'. It details a merge pull request #9 from 'codefresh-io' to 'master'. The pipeline is visualized as a sequence of stages: Development (non-production), Staging (production), US-East (production), EU (production), and Asia (production). Each stage lists its steps and their durations. The EU stage is currently in progress and shows a failure in the 'e2e\_test' step, indicated by a red 'X' and a count of 4 issues. Other stages like Staging and US-East show successful completion of their respective steps. The interface also includes a search bar, a 'Quick Search' field, and a user profile for 'idanarbel codefresh-inc'.



# Topics

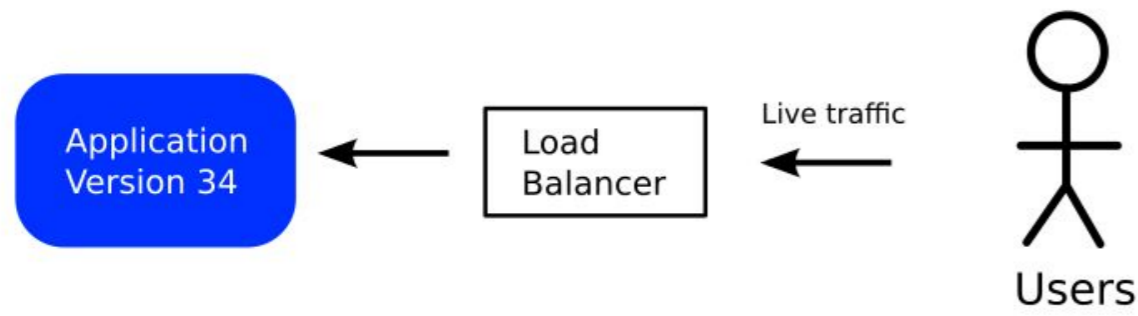
- 1 **Argo Rollouts Intro**
- 2 **Use cases with huge blast radius**
- 3 **The new Kubernetes Gateway API**
- 4 **Static routing (Demo)**
- 5 **Dynamic routing (Demo)**
- 6 **Comparison/Discussion**



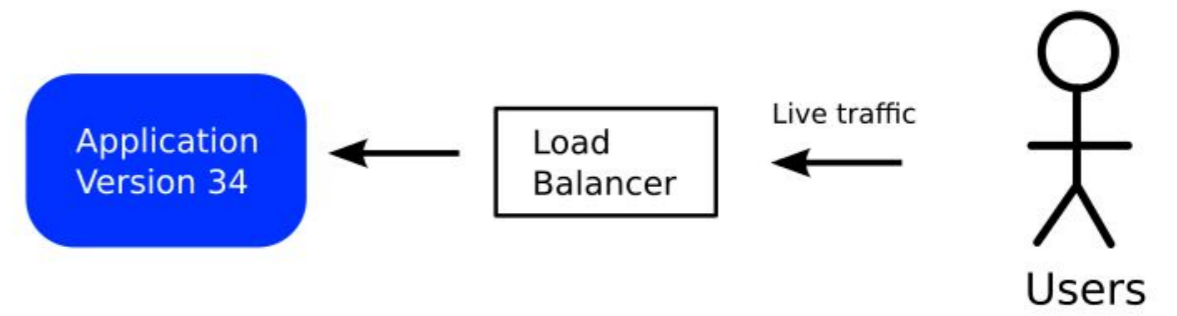
# Progressive Delivery



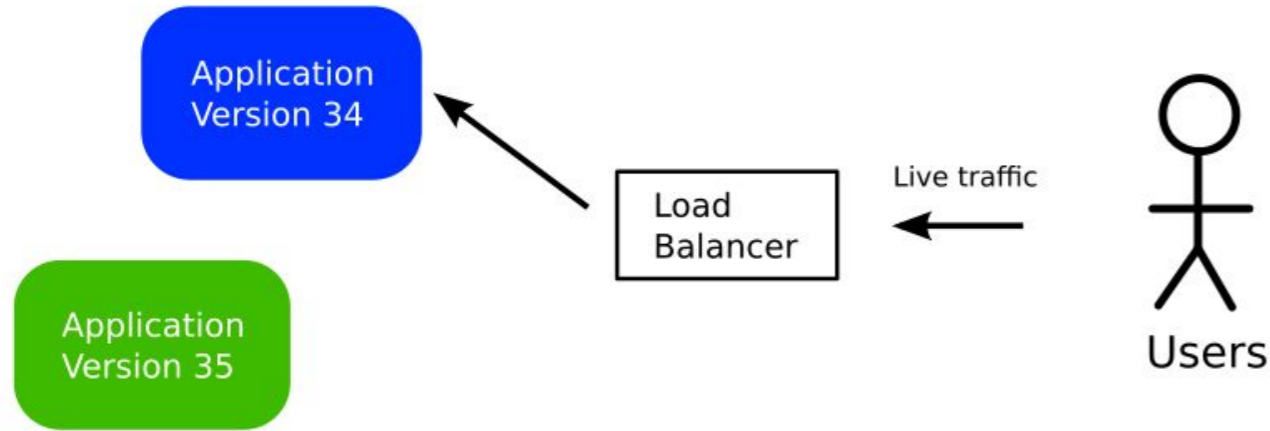
### 1- Initial version



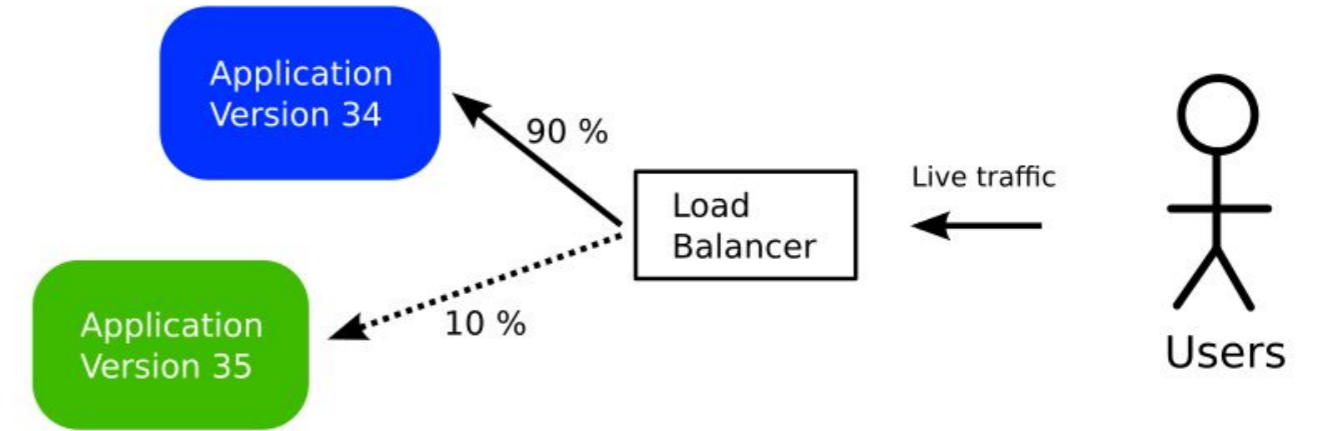
### 1- Initial version



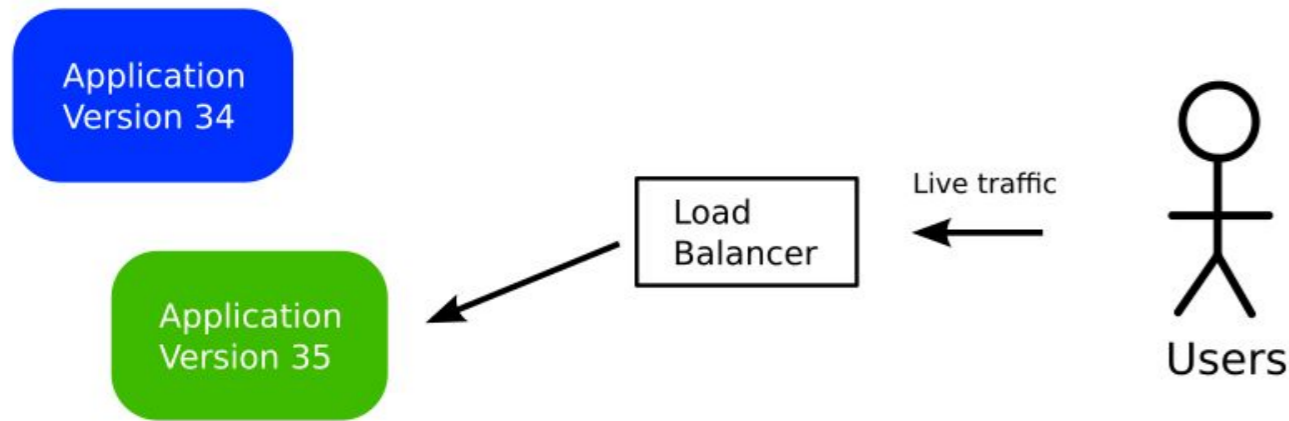
### 2- New version deployed



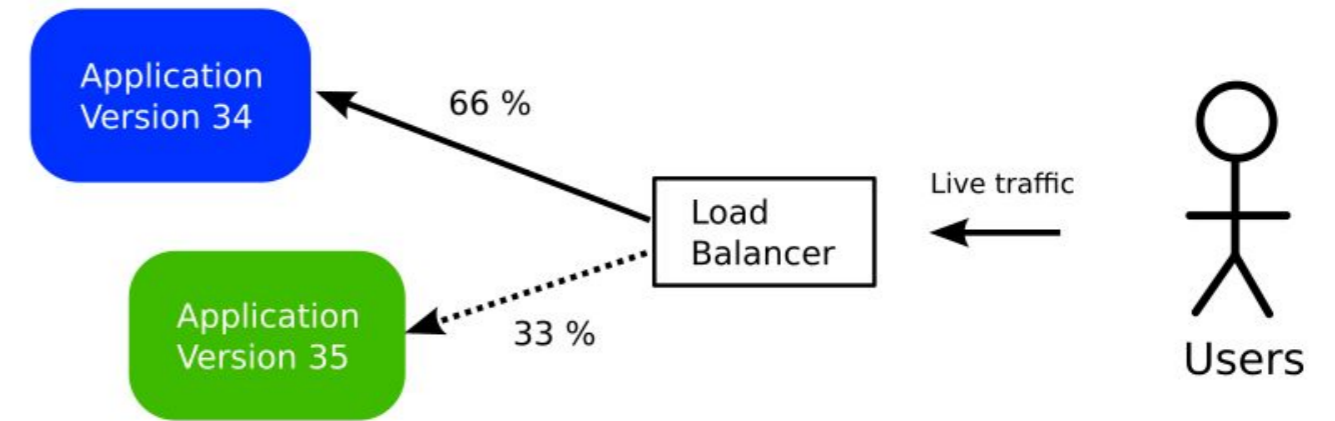
### 2- New version used by 10% of users



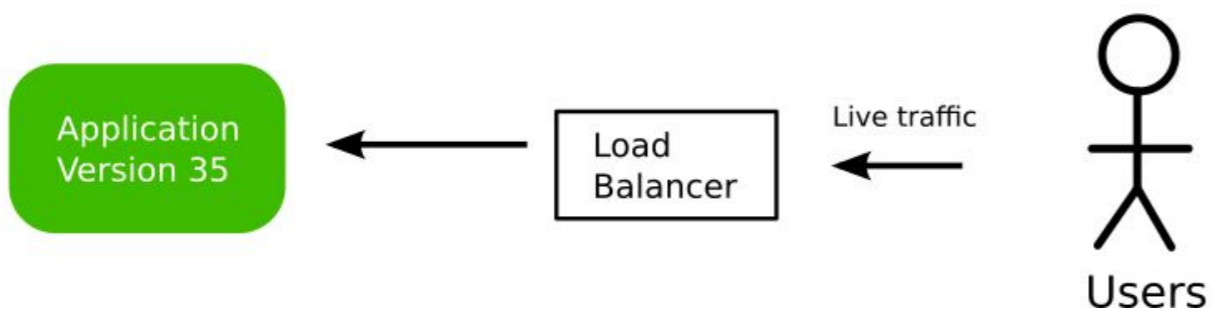
### 3- Switch Traffic



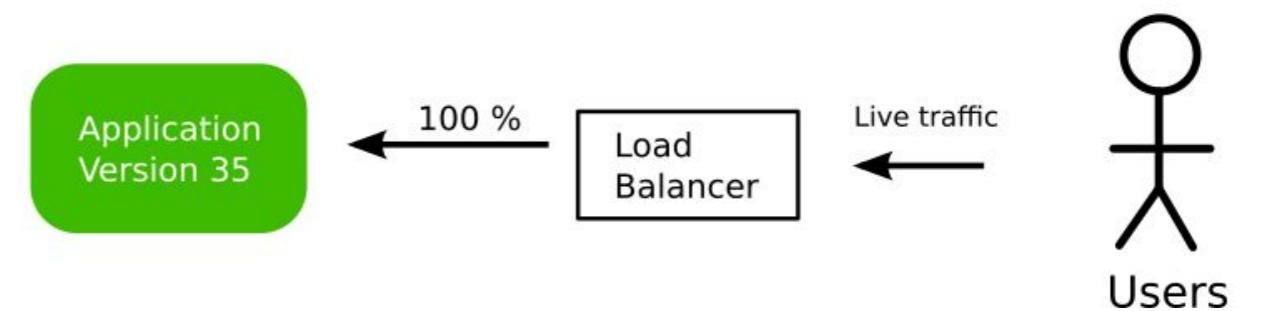
### 3- New version used by 33% of users



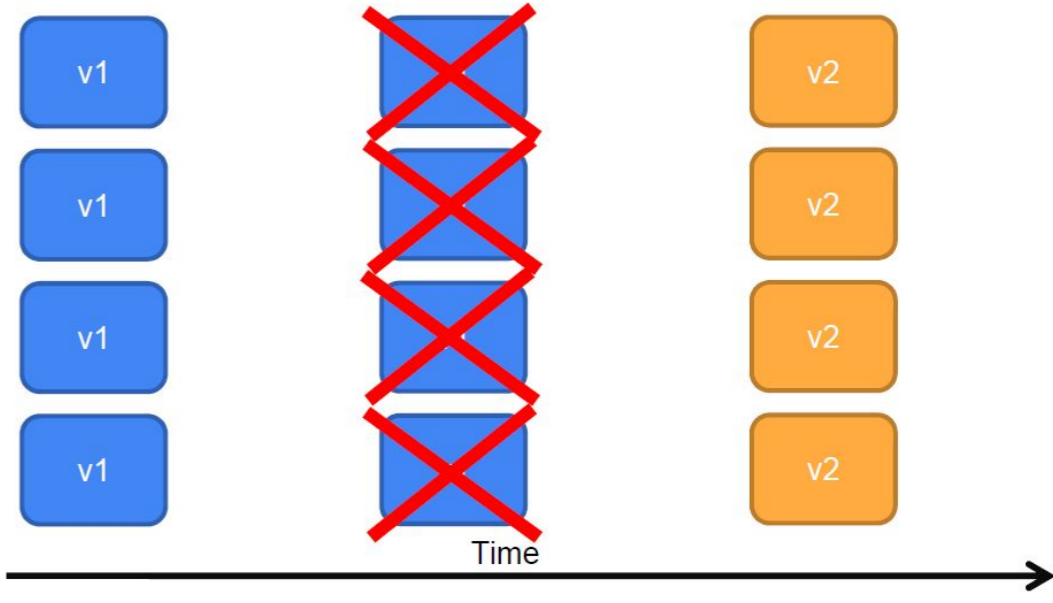
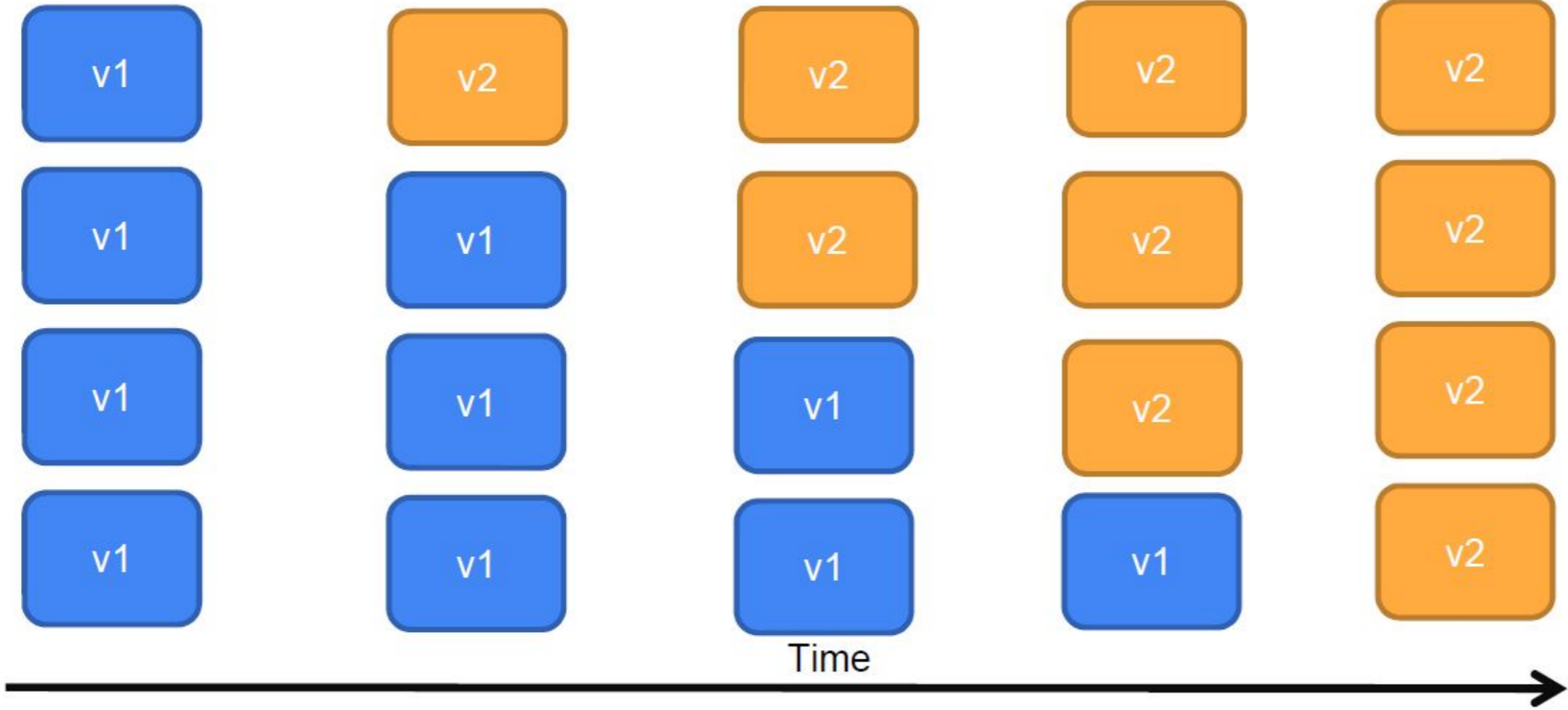
### 4- Finish



### 4- New version is used by all users



# Default Kubernetes deployments



# Canaries with Kubernetes







# Argo Rollouts

  2405

Advanced Kubernetes deployment strategies such as Canary and Blue-Green made easy.

[Learn More](#)

<https://argoproj.github.io/>



# Argo Rollouts









- Rollouts (new CRD)
- Extends Deployment
- Blue/Green&Canaries
- Minimal dashboard
- Pre/Post checks

The screenshot shows the Argo Rollouts dashboard for a rollout named "rollouts-demo". At the top right, there are control buttons: Restart, Retry, Abort, Promote, and PromoteFull. The main content is divided into several panels:

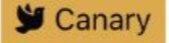
- Steps:** A vertical sequence of steps: Set Weight: 20%, Pause, Set Weight: 40%, Pause: 10s, Set Weight: 60%, Pause: 10s, Set Weight: 80%, and Pause: 10s. The "Pause" step is currently active and highlighted with a red border.
- Summary:** Shows the rollout strategy as "Canary", the current step as "1/8", and the current set weight and actual weight as "20".
- Containers:** Shows the rollout name "rollouts-demo" and the current container image "argoproj/rollouts-demo:yellow".
- Revisions:** Shows two revisions:
  - Revision 2:** "argoproj/rollouts-demo:yellow" with revision ID "rollouts-demo-6cf78c66c5". It is marked as "canary" and has a green checkmark.
  - Revision 1:** "argoproj/rollouts-demo:blue" with revision ID "rollouts-demo-687d76d795". It is marked as "stable" and has five green checkmarks. A "Rollback" button is visible next to it.





### Steps


-  Set Weight: 20%
-  Pause
-  Set Weight: 40%
-  Pause: 10s
-  Set Weight: 60%
-  Pause: 10s
-  Set Weight: 80%
-  Pause: 10s

### Summary


Strategy  Canary

Step  1/8

Set Weight  20


Actual Weight  20

### Containers

 Edit

rollouts-demo



argoproj/rollouts-demo:yellow




### Revisions

#### Revision 2



argoproj/rollouts-demo:yellow






rollouts-demo-6cf78c66c5  canary 



#### Revision 1

argoproj/rollouts-demo:blue

rollouts-demo-687d76d795  Rollback 



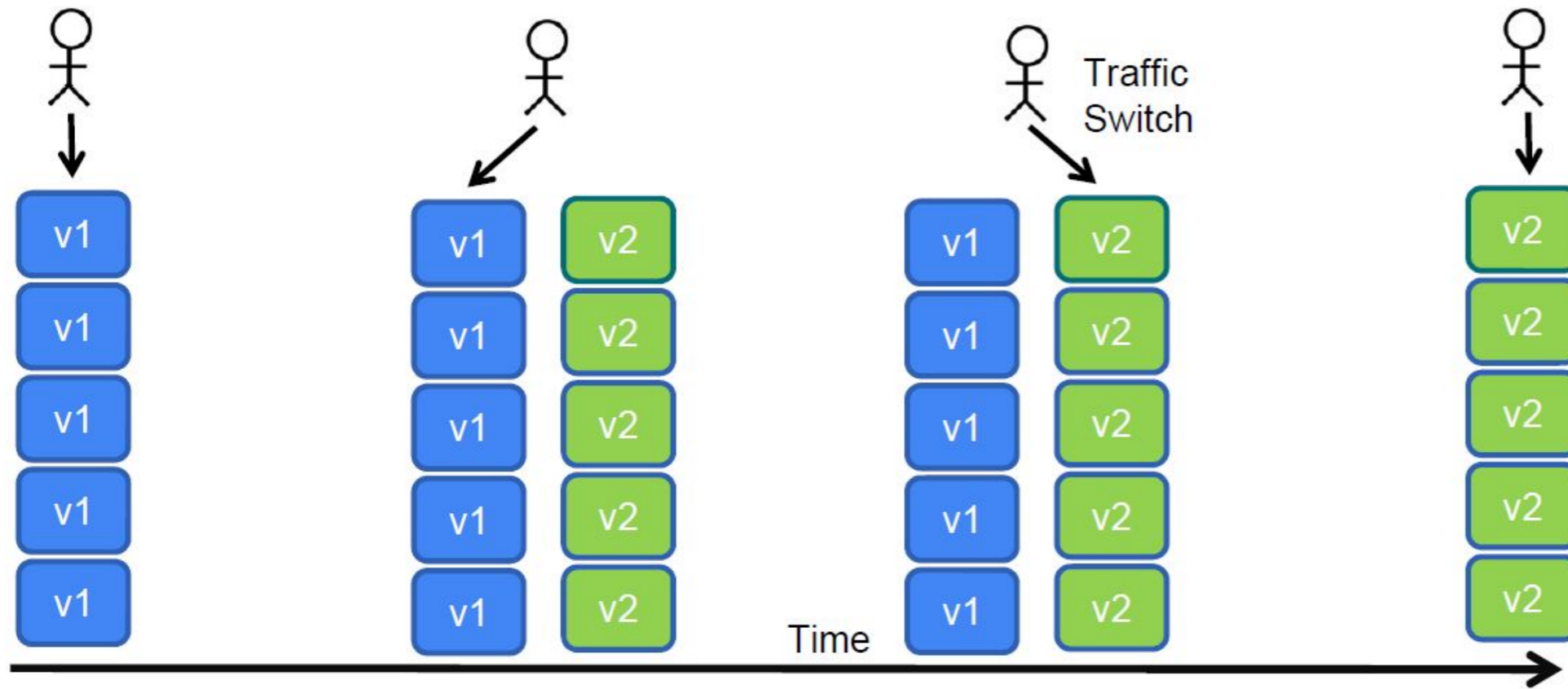
```
apiVersion: argoproj.io/v1alpha1
kind: Rollout
metadata:
  name: example-rollout
spec:
  replicas: 10
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:1.15.4
          ports:
            - containerPort: 80
  minReadySeconds: 30
  revisionHistoryLimit: 3
```

```
strategy:
  canary: #Indicates that the rollout should use the Canary strategy
  maxSurge: "25%"
  maxUnavailable: 0
  steps:
    - setWeight: 10
    - pause:
        duration: 1h # 1 hour
    - setWeight: 20
    - pause: {} # pause indefinitely
```

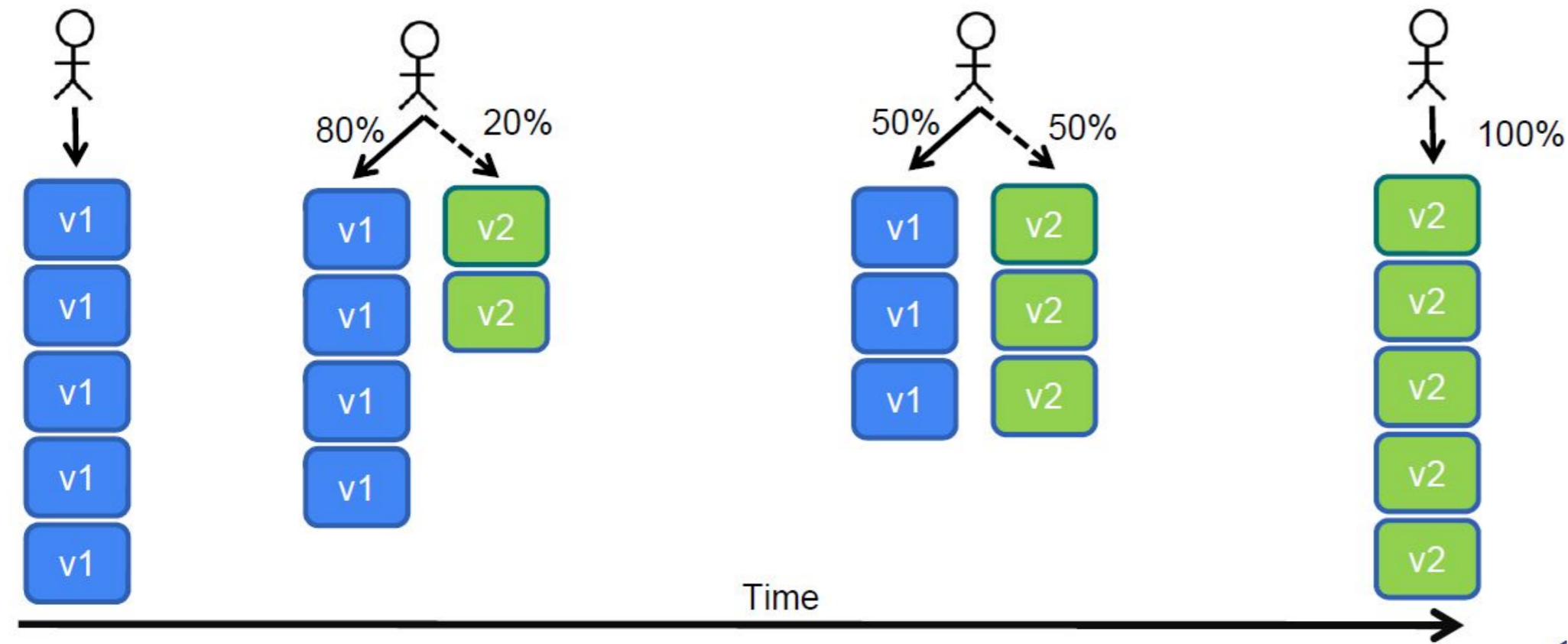
Strategy

# Rollout extends K8s deployment





# Kubernetes Progressive Delivery



# Without/With traffic management

Prod  
50%



Canary  
50%



Linkerd



Prod  
95%



Canary  
5%



Prod  
75%



Canary  
25%



Linkerd



Prod  
70%



Canary  
30%



Prod  
90%



Canary  
10%



Linkerd



Prod  
20%



Canary  
80%



# Supported Traffic managers

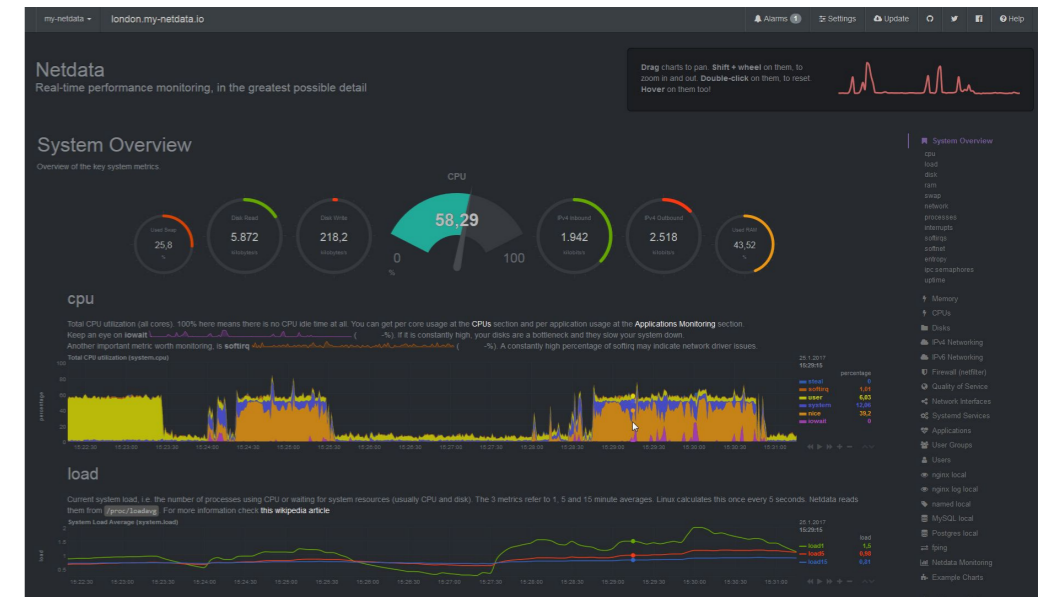
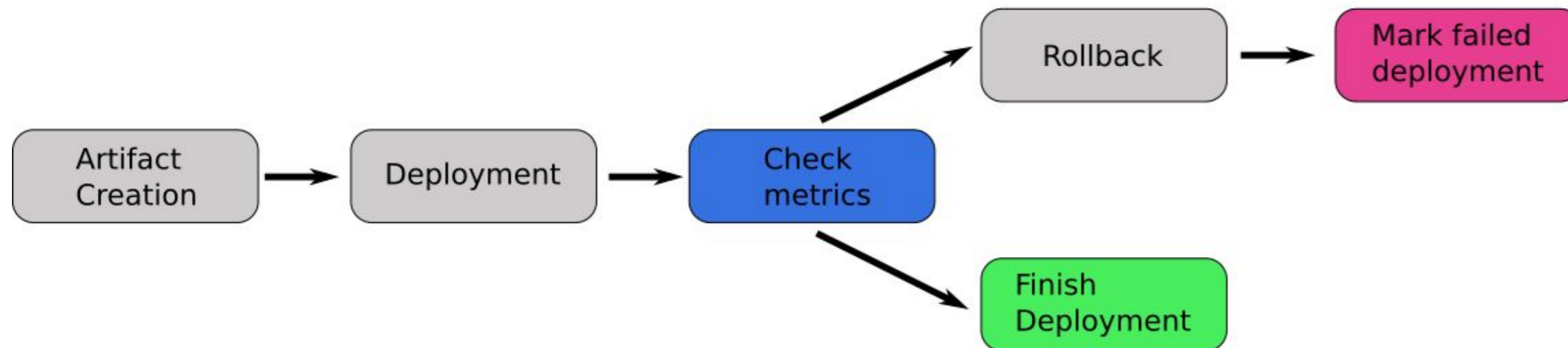
- AWS Ingress Controller
- Ambassador Labs
- Apache APISIX
- Linkerd
- Istio
- Kong
- Nginx
- Traefik
- Openshift Routes
- Gloo Gateway
- Contour
- Cilium
- Envoy Gateway
- Gateway API



# Pre/Post checks



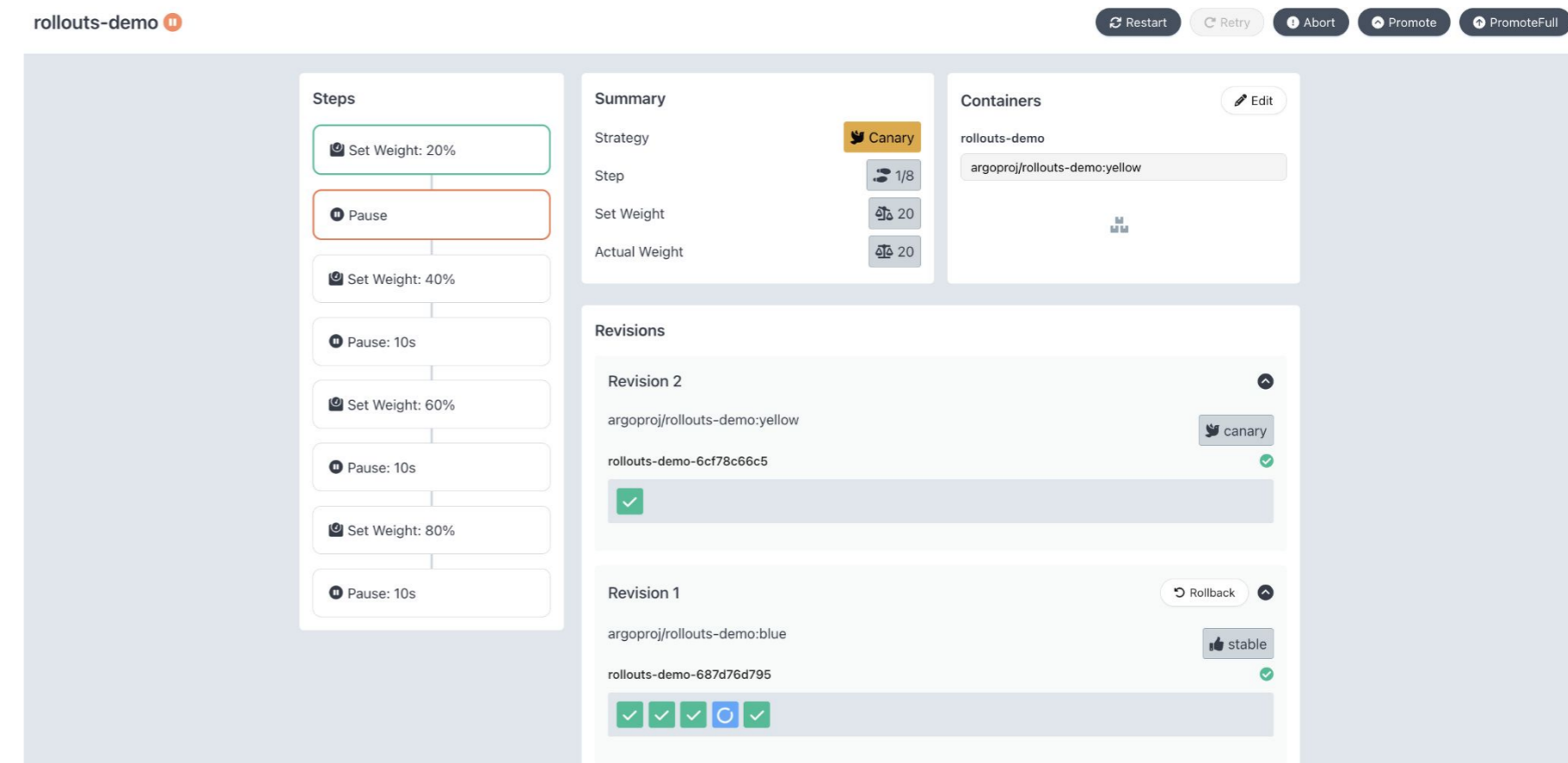
## Fully Automated Rollbacks





# Argo Rollouts - Other features

- A/B testing
- Header based routing
- Argo CD UI extension
- Notifications
- Plugins
- CLI/Metrics



The screenshot displays the Argo Rollouts UI for a resource named 'rollouts-demo'. At the top right, there are control buttons: Restart, Retry, Abort, Promote, and PromoteFull. The main interface is divided into several panels:

- Steps:** A vertical sequence of steps for a canary rollout:
  - Set Weight: 20% (green border)
  - Pause (orange border)
  - Set Weight: 40%
  - Pause: 10s
  - Set Weight: 60%
  - Pause: 10s
  - Set Weight: 80%
  - Pause: 10s
- Summary:** Shows the current strategy as 'Canary', 1/8 steps completed, and a set weight of 20%. The actual weight is also shown as 20%.
- Containers:** Lists the containers for the rollout: 'rollouts-demo' and 'argoproj/rollouts-demo:yellow'.
- Revisions:** Shows two revisions:
  - Revision 2:** 'argoproj/rollouts-demo:yellow' with ID 'rollouts-demo-6cf78c66c5'. It is in a 'canary' state and has a green checkmark.
  - Revision 1:** 'argoproj/rollouts-demo:blue' with ID 'rollouts-demo-687d76d795'. It is in a 'stable' state and has five green checkmarks. A 'Rollback' button is visible next to it.

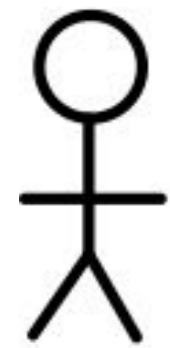


# Use Cases



# “Typical” Use case

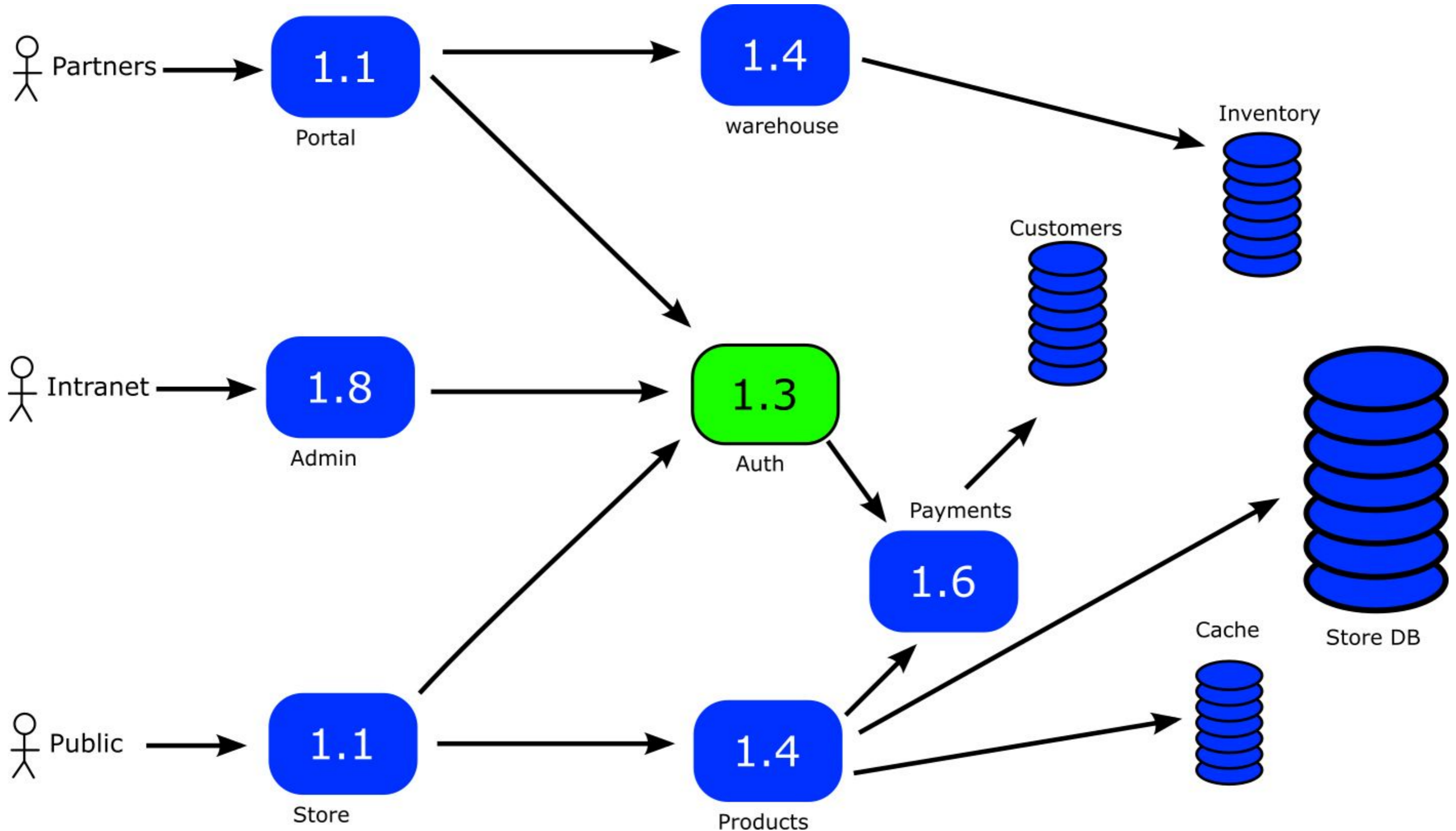
Single Service/Application



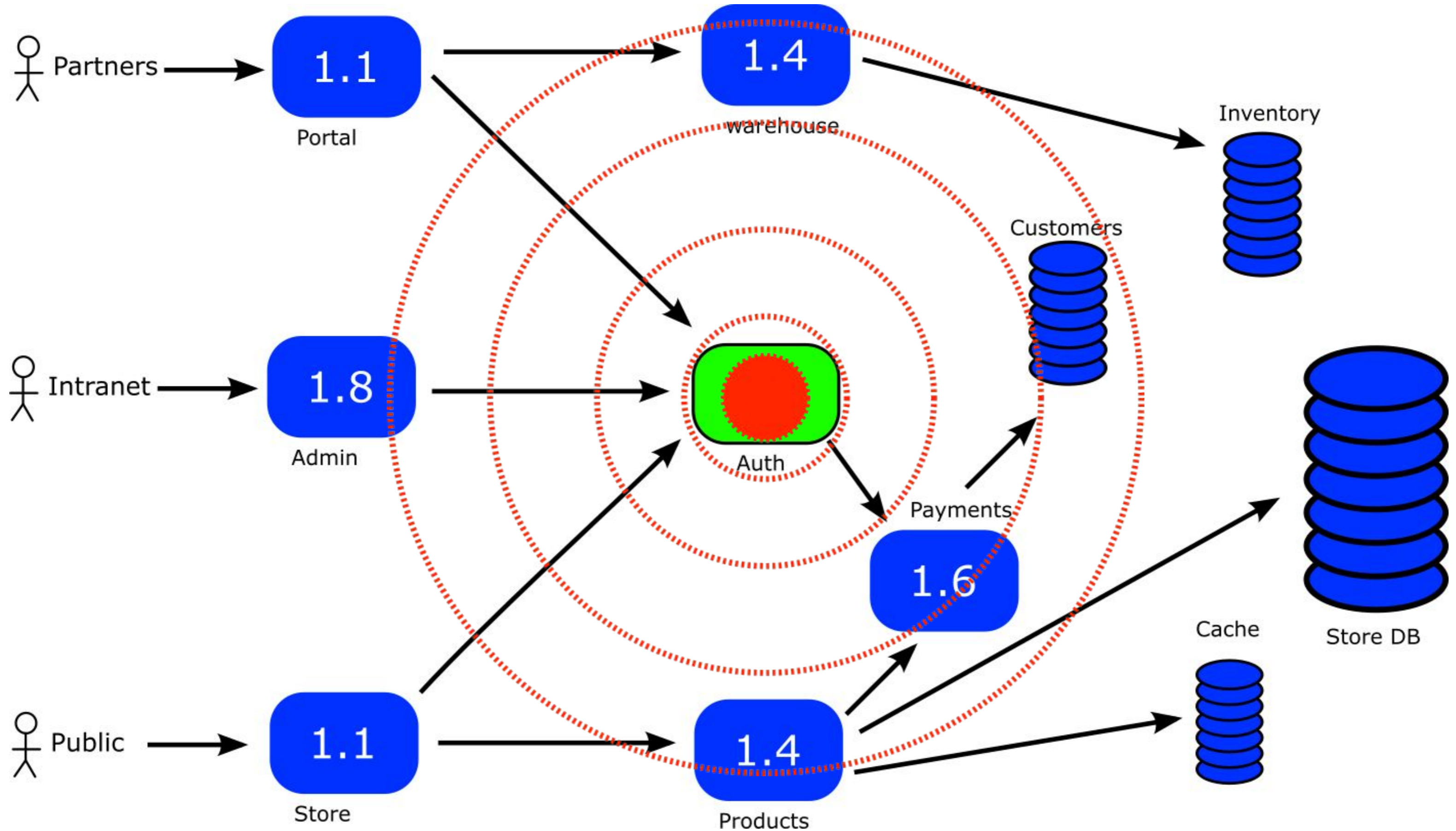
Traffic



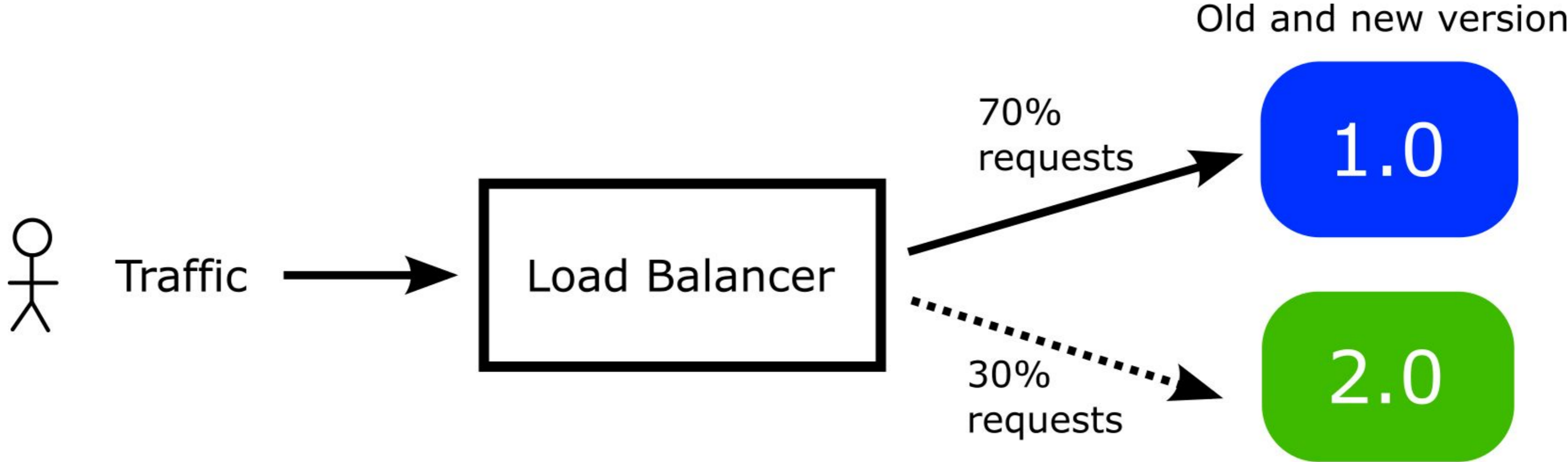
# What happens in the real world



# Huge blast radius



# Canary connections are RANDOM by default

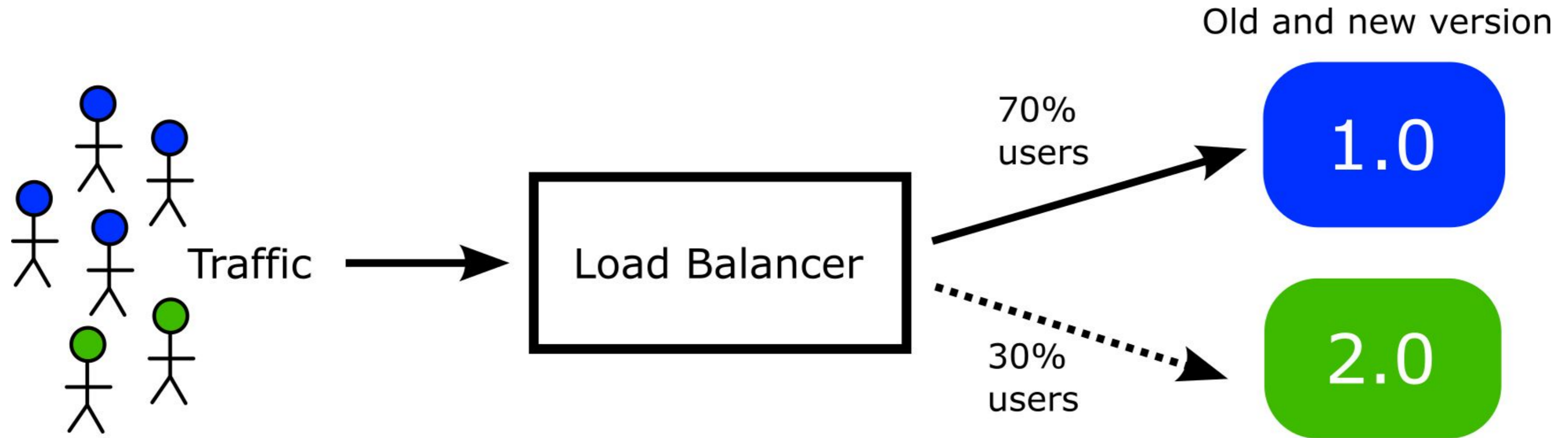


- Request 1 (old)
- Request 2 (new)
- Request 3 (old)
- Request 4 (old)
- Request 5 (new)

**REJECTED**



# We want canaries on users (not requests)



Request 1 (old)  
Request 2 (old)  
Request 3 (old)  
Request 4 (old)  
Request 5 (old)

Request 1 (new)  
Request 2 (new)  
Request 3 (new)  
Request 4 (new)  
Request 5 (new)



# We want to control who sees the canary

“Only our internal users should see this new version”

“Only French users must be part of the canary”

“Asia should stay in old version, US will see the canary only”

“Only users that have checked the ‘preview checkbox’ must see the canary”

“The payment gateway should stay in the old version. The intranet should see the new version”





# What you get with vanilla Argo Rollouts

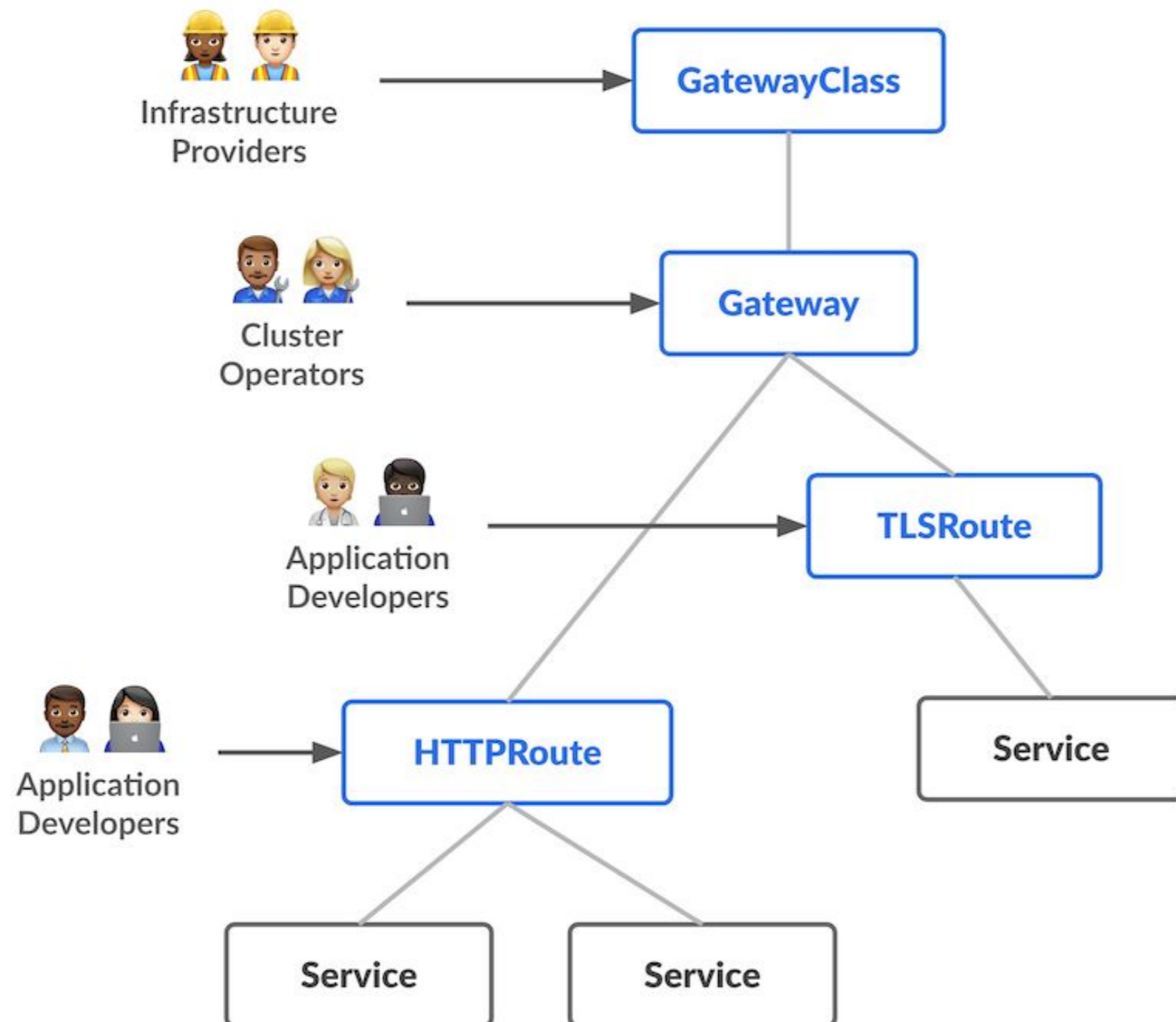
- ✓ Native Resource for Canary Deployments in Kubernetes
- ✗ Huge blast radius if something goes wrong
- ✗ Blast radius applies to all potential users
- ✗ Canary percentages are request based and not user based
- ✗ Canary requests are random by default
- ✗ Single user can see preview/stable requests in the same application
- ✗ No control over which users see the canary



# Intermission: The Gateway API



# Gateway API is Ingress Next Gen



<https://gateway-api.sigs.k8s.io/>

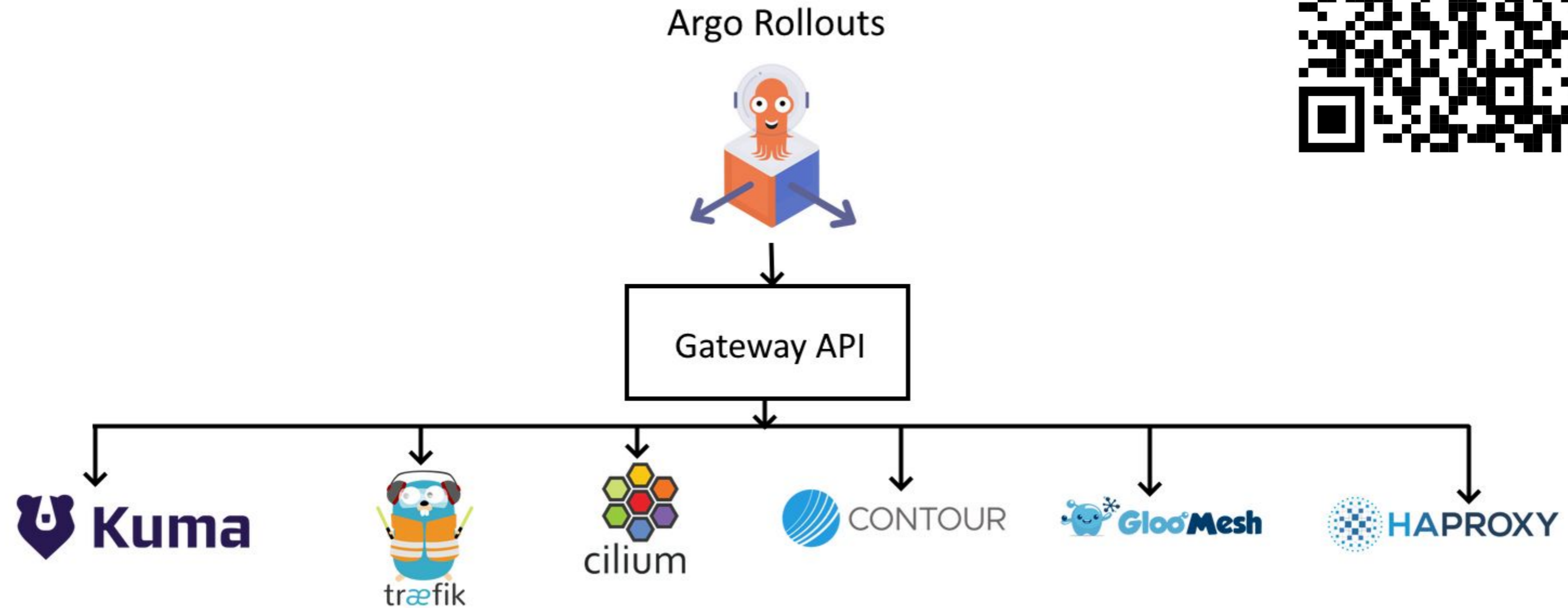


# Supported Traffic managers

- AWS Ingress Controller
- Ambassador Labs
- Apache APISIX
- Linkerd
- Istio
- Kong
- Nginx
- Traefik
- Openshift Routes
- Gloo Gateway
- Contour
- Cilium
- Envoy Gateway
- Gateway API



# One API to rule them all

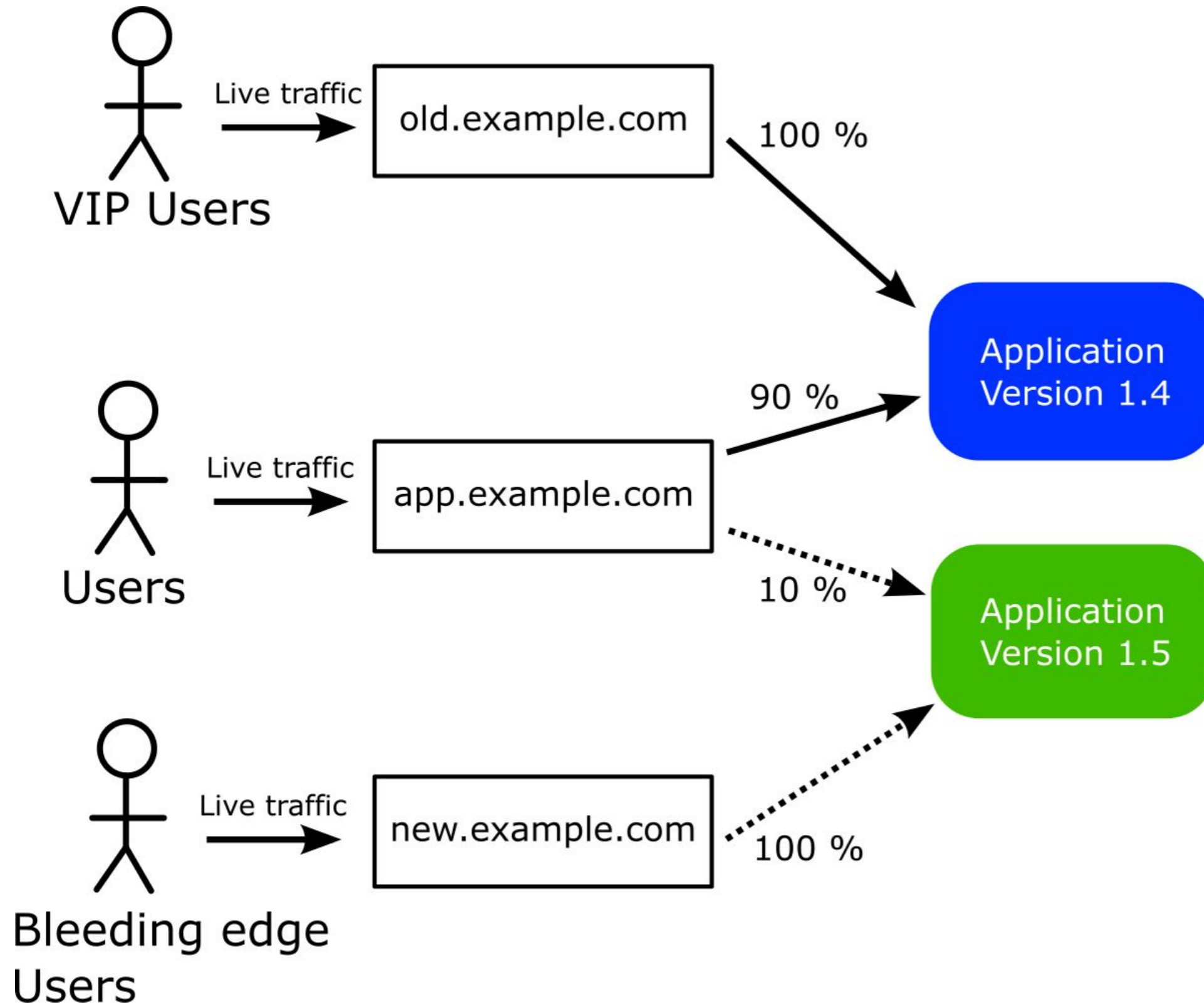


<https://github.com/argoproj-labs/rollouts-plugin-trafficrouter-gatewayapi>



# Static URL routing

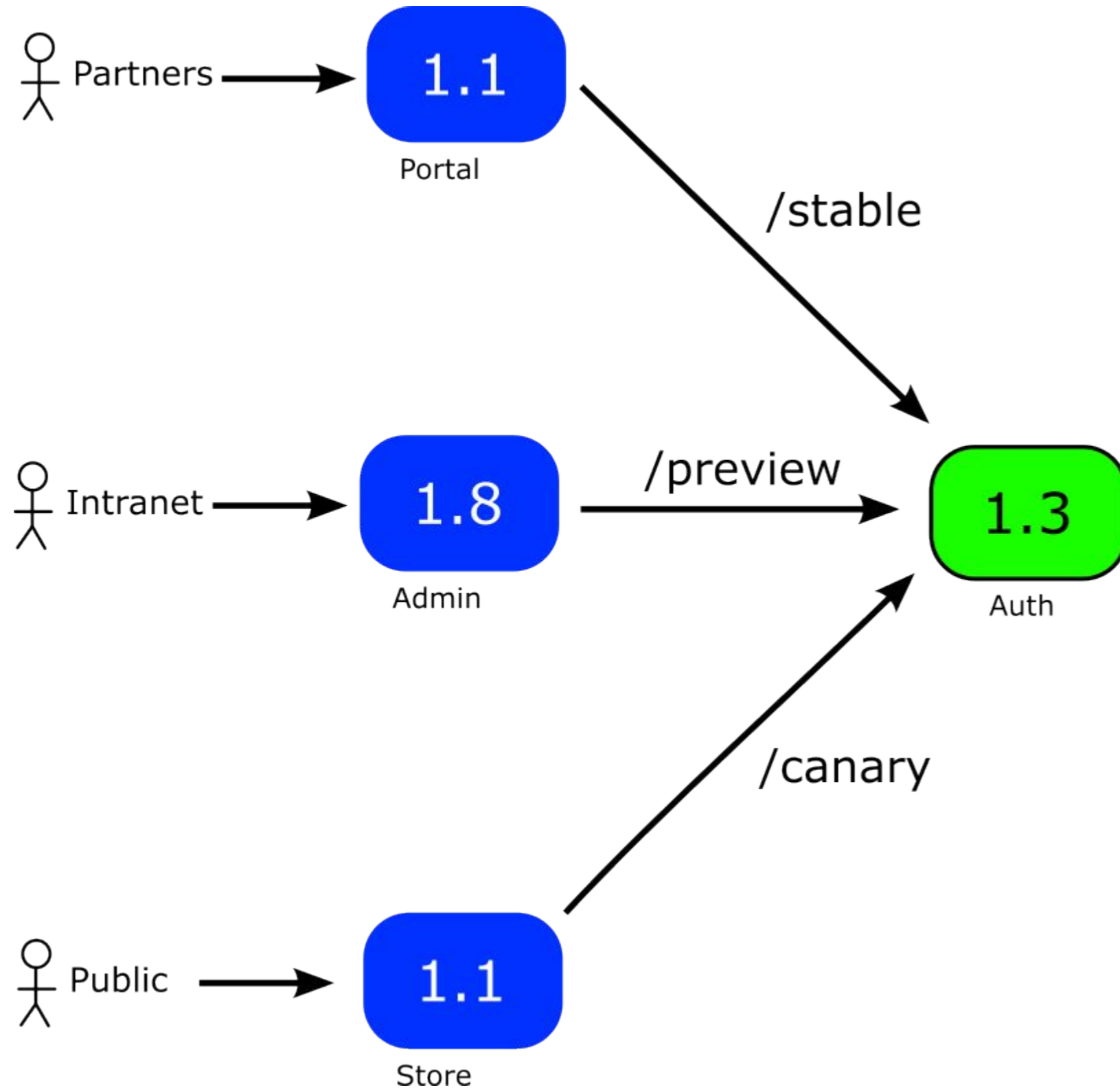




**Use different URLs  
for each category  
of users**



# Host or path based routing



- /stable, /preview, /canary

- canary.auth.com
- preview.auth.com
- stable.auth.com

- partners.acme.inc
- intranet.acme.inc
- public.acme.inc





Canary Start

Canary End

/stable



/preview



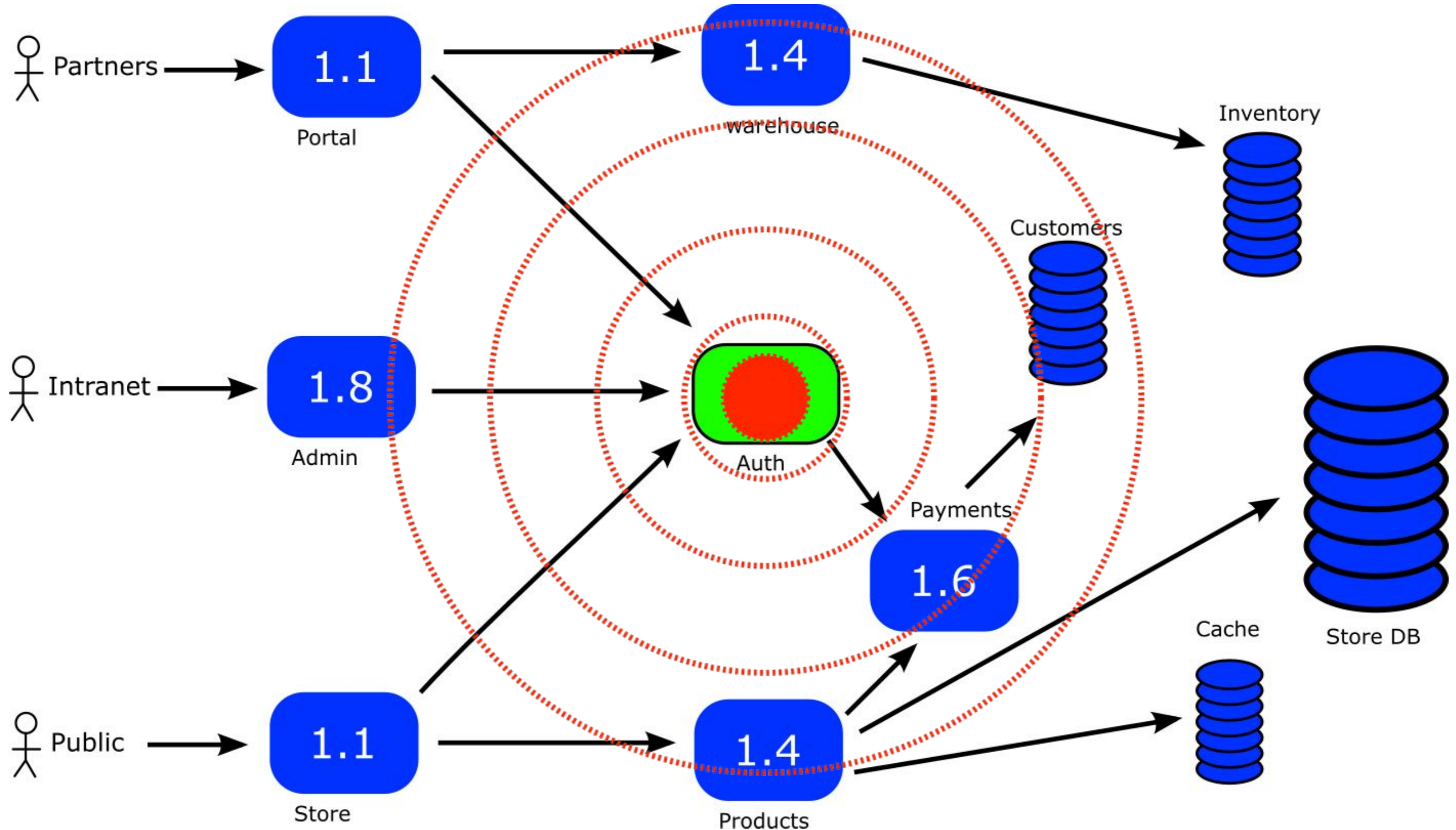
/canary



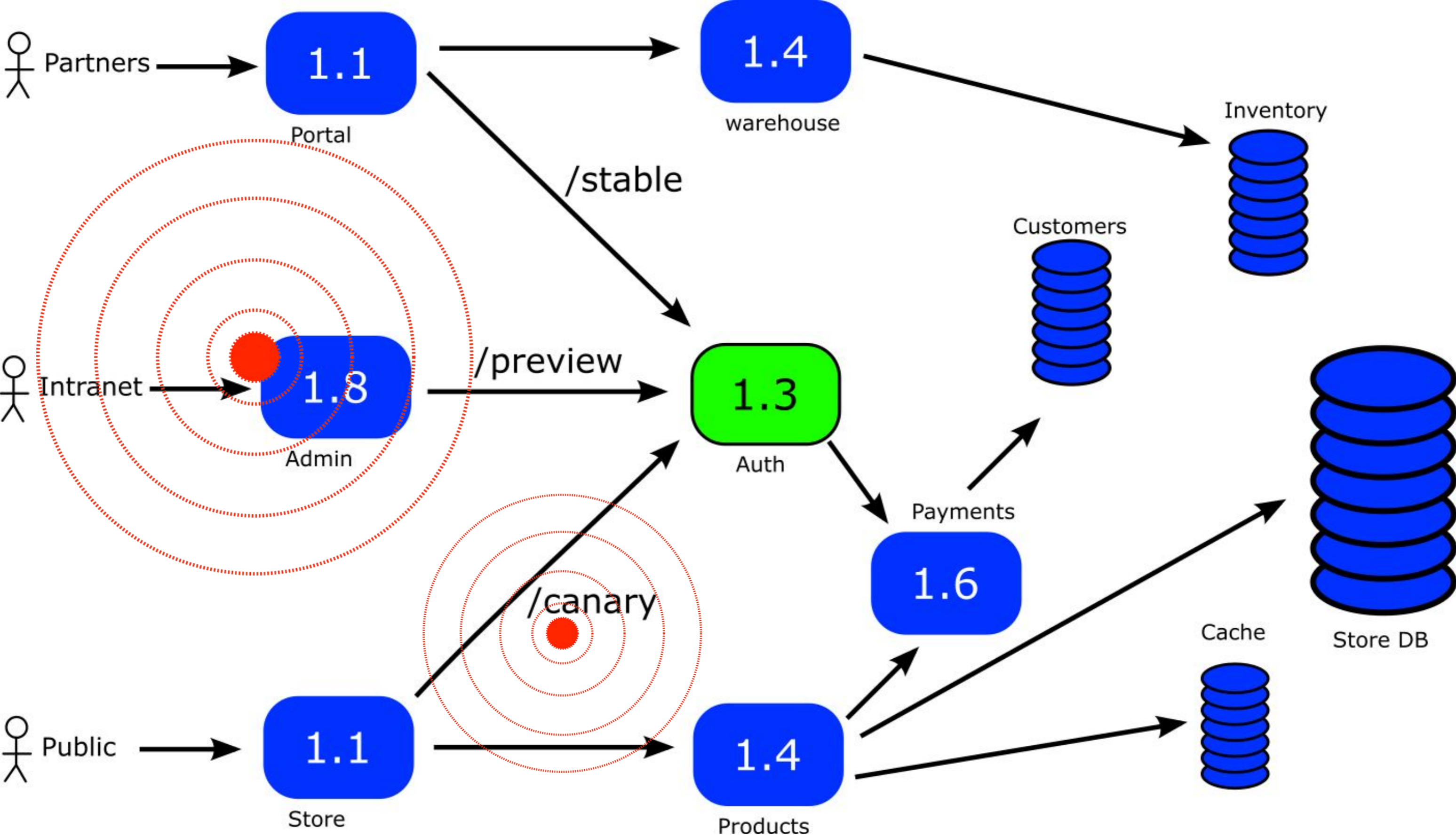
Time



# Before (everybody is affected)



# After (full control over impact)



# How it works

```
---
kind: HTTPRoute
apiVersion: gateway.networking.k8s.io/v1beta1
metadata:
  name: canary-route
  namespace: default
spec:
  parentRefs:
  - name: eg
  hostnames:
  - app.example.com
  rules:
  - matches:
    - path:
      type: PathPrefix
      value: /
    backendRefs:
    - name: argo-rollouts-stable-service
      kind: Service
      port: 80
    - name: argo-rollouts-canary-service
      kind: Service
      port: 80
```

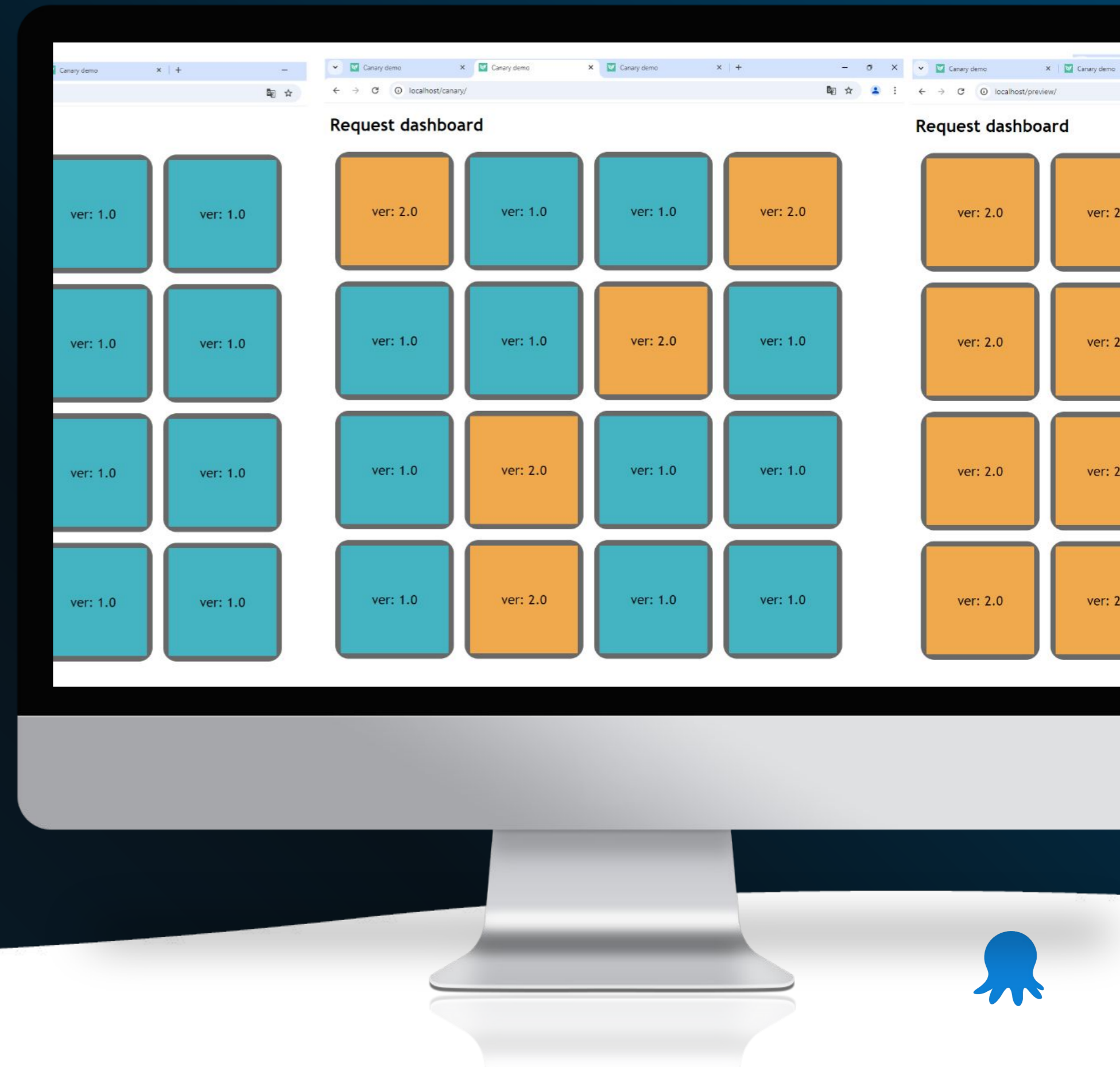
```
---
kind: HTTPRoute
apiVersion: gateway.networking.k8s.io/v1beta1
metadata:
  name: always-old-version
  namespace: default
spec:
  parentRefs:
  - name: eg
  hostnames:
  - old.example.com
  rules:
  - matches:
    - path:
      type: PathPrefix
      value: /
    backendRefs:
    - name: argo-rollouts-stable-service
      kind: Service
      port: 80
```

```
---
kind: HTTPRoute
apiVersion: gateway.networking.k8s.io/v1beta1
metadata:
  name: always-new-version
  namespace: default
spec:
  parentRefs:
  - name: eg
  hostnames:
  - new.example.com
  rules:
  - matches:
    - path:
      type: PathPrefix
      value: /
    backendRefs:
    - name: argo-rollouts-canary-service
      kind: Service
      port: 80
```



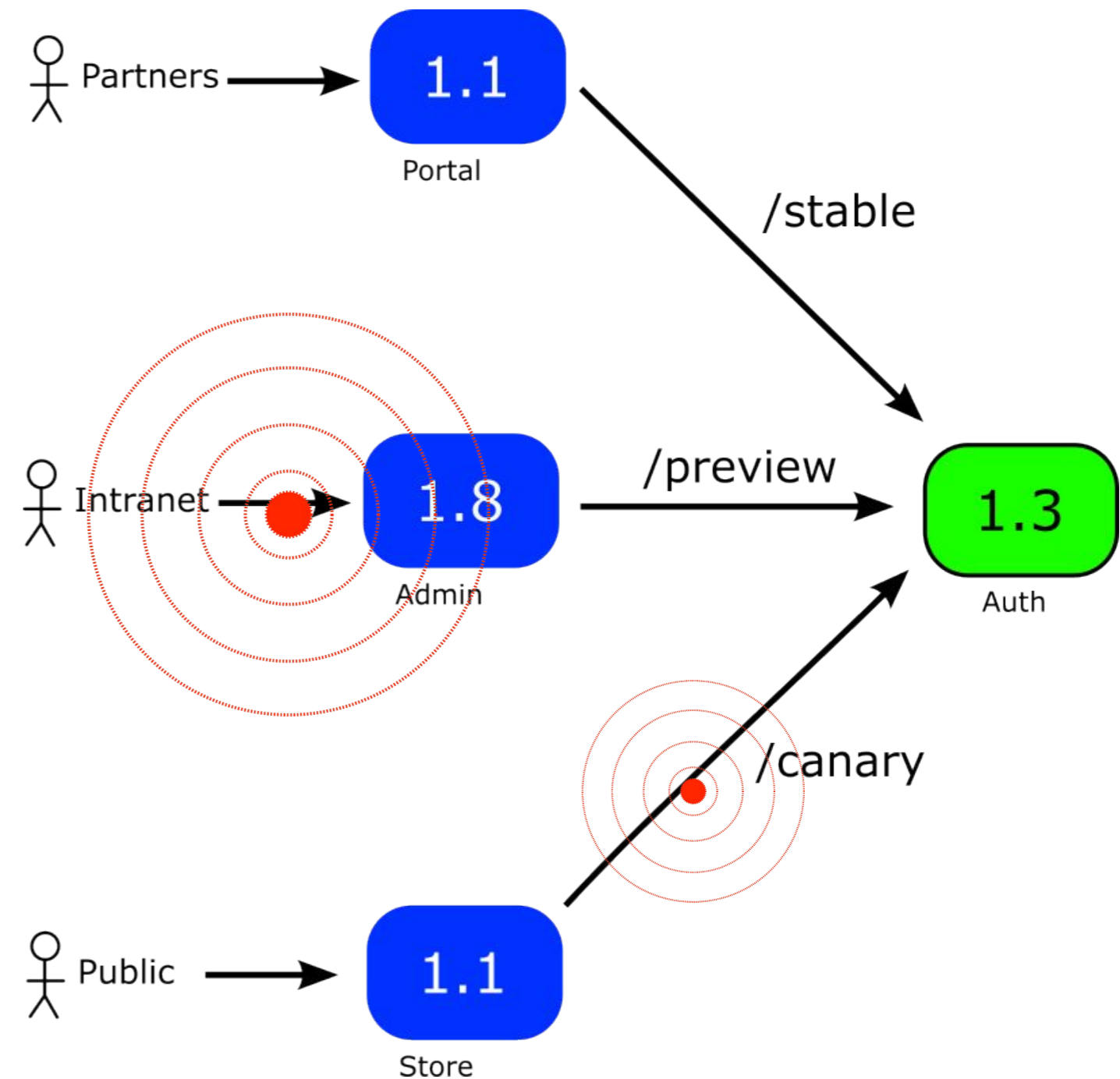
# Static Routing

Live Demo with 3 different endpoints  
for stable/canary/preview



# Static routing pros/cons

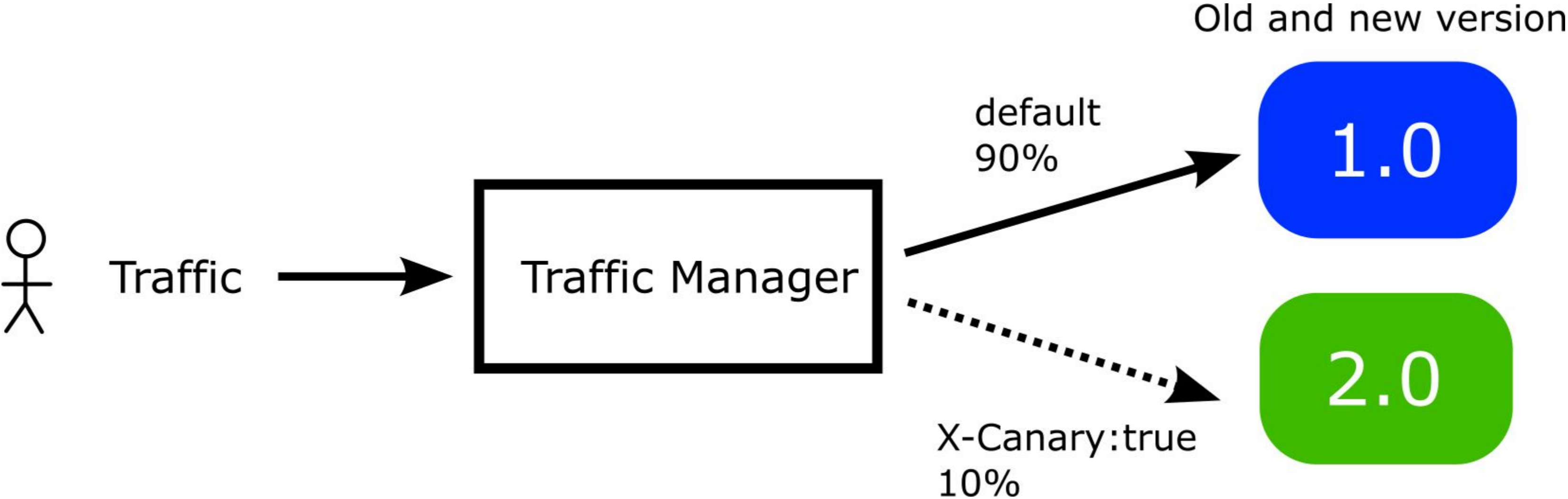
- ✓ Full Control of blast radius
- ✓ Clients can choose their risk (in advance)
- ✗ Setup is fixed and hard to change
- ✗ Requests within canary are still random
- ✗ No ability to start canary on specific regions, or user groups
- ✗ Needs static configuration changes
- ✗ Cannot change canary users on the fly



# Dynamic routing

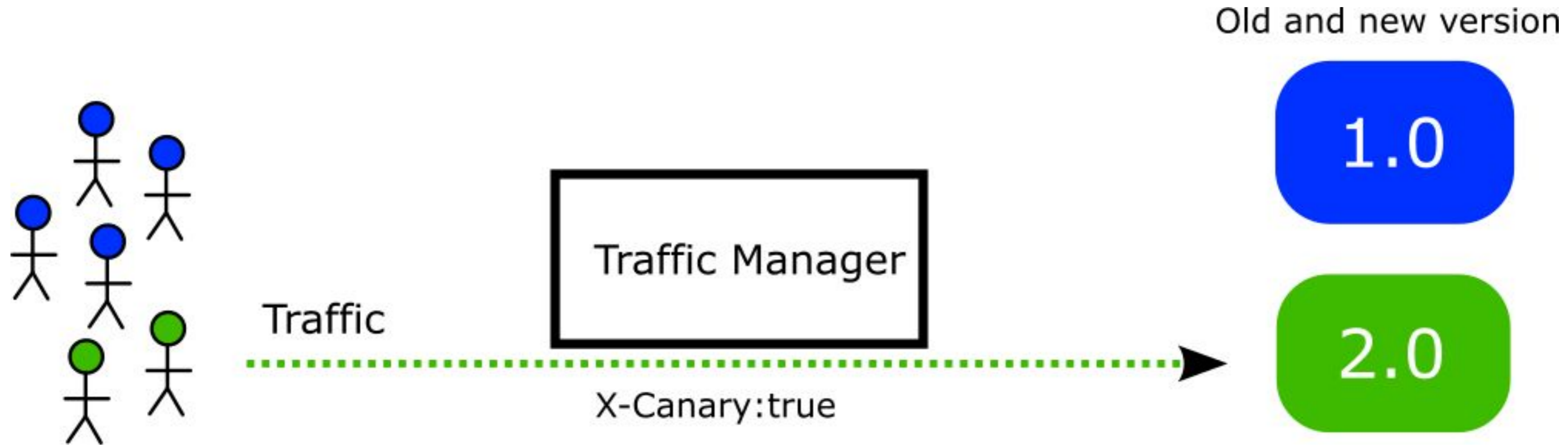


# Canaries with custom HTTP headers





# Finally we have canary per user



# HTTP headers are fully dynamic

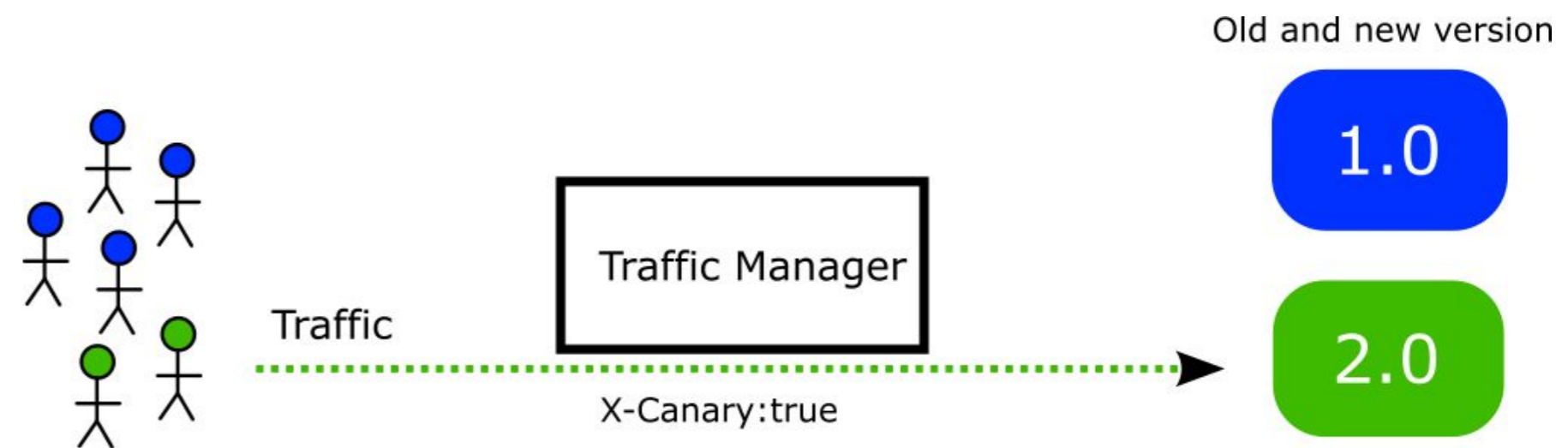
Headers can change on the fly by the app

Proxies/Load Balancers can inject/change headers

No static configuration any more

Canary users can change per release

User can change their risk acceptance in the middle of a canary if they wish



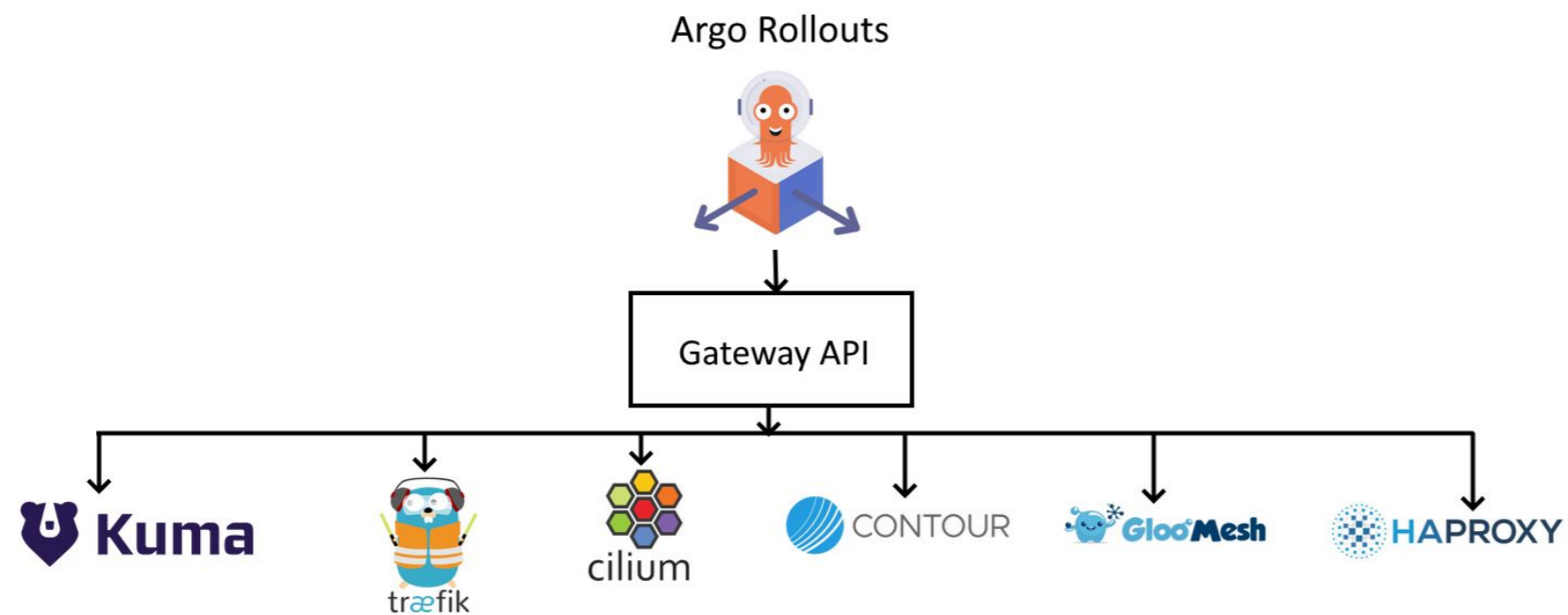
```
apiVersion: argoproj.io/v1alpha1
kind: Rollout
metadata:
  name: smart-rollouts-demo
spec:
  replicas: 5
  strategy:
    canary:
      canaryService: smart-canary-service
      stableService: smart-stable-service
      trafficRouting:
        managedRoutes:
          - name: always-preview
        plugins:
          argoproj-labs/gatewayAPI:
            httpRoutes:
              - name: my-smart-route
                useHeaderRoutes: true
            namespace: default
      steps:
        - setHeaderRoute:
            name: always-preview
            match:
              - headerName: X-Canary
                headerValue:
                  exact: "yes"
        - setWeight: 25
        - pause: {}
        - setWeight: 100
```

```
apiVersion: gateway.networking.k8s.io/v1
kind: HTTPRoute
metadata:
  name: my-smart-route
spec:
  parentRefs:
    - name: traefik-gateway
      namespace: default
  rules:
    - matches:
        - path:
            type: PathPrefix
            value: /
      backendRefs:
        - name: smart-stable-service
          kind: Service
          port: 80
        - name: smart-canary-service
          kind: Service
          port: 80
```



# Check your traffic provider first

- The Argo Rollouts plugin fully supports header based routing
- Gateway API is just a specification. The provider needs to support it as well
- Previously this capability was only available to Istio
- Now it is available everywhere where the provider supports the spec



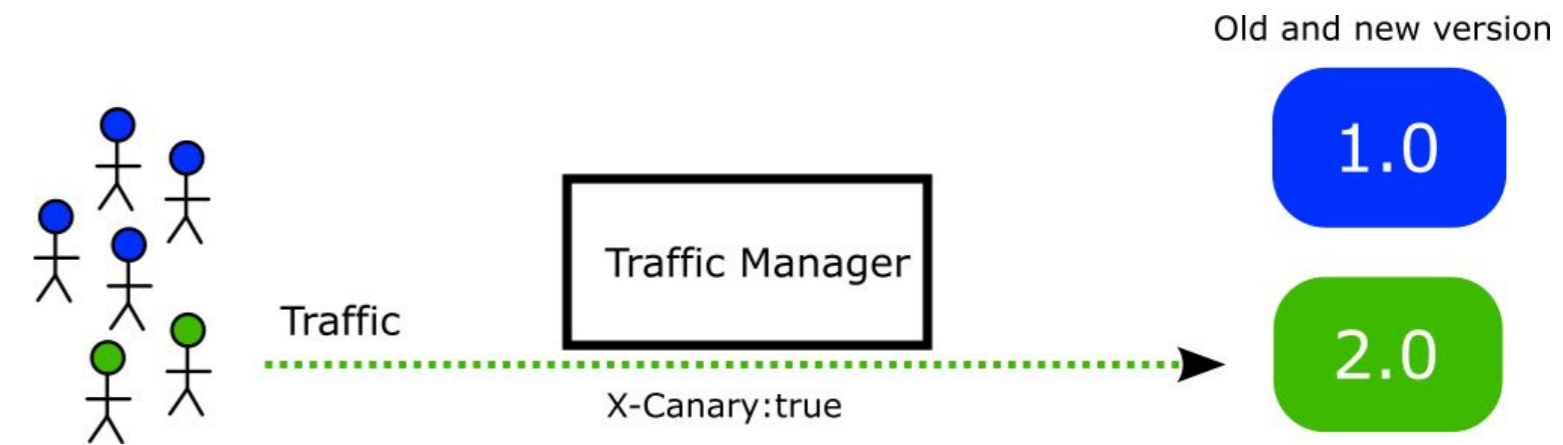
# Dynamic Routing

Live Demo with header based canary selection



# Dynamic routing pros/cons

- ✓ Full Control of blast radius
- ✓ Clients can choose their risk at will
- ✓ Setup is flexible and easy to change
- ✓ Canary is per user (and per request)
- ✓ Ability to start canary on specific regions, or user groups
- ✓ Can change canary users on the fly
- ✗ Might need source code changes



# Conclusion

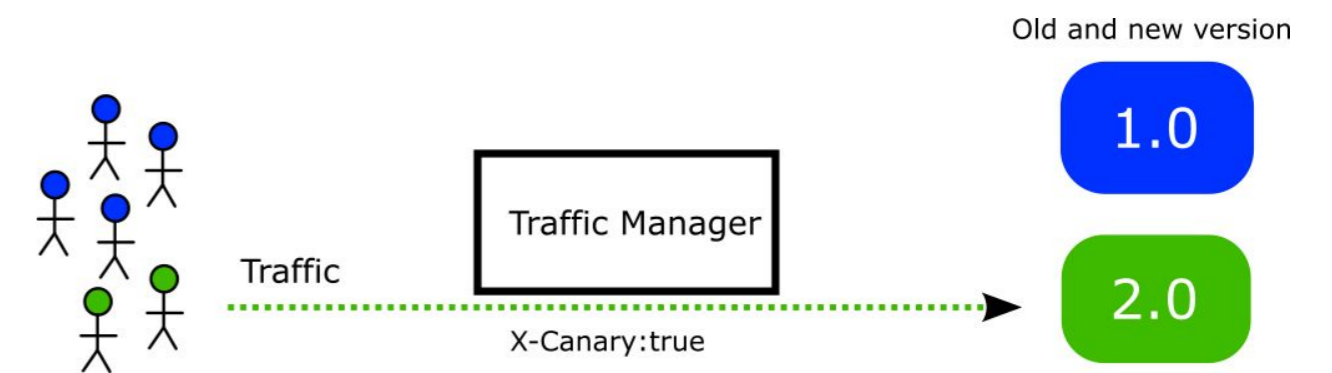
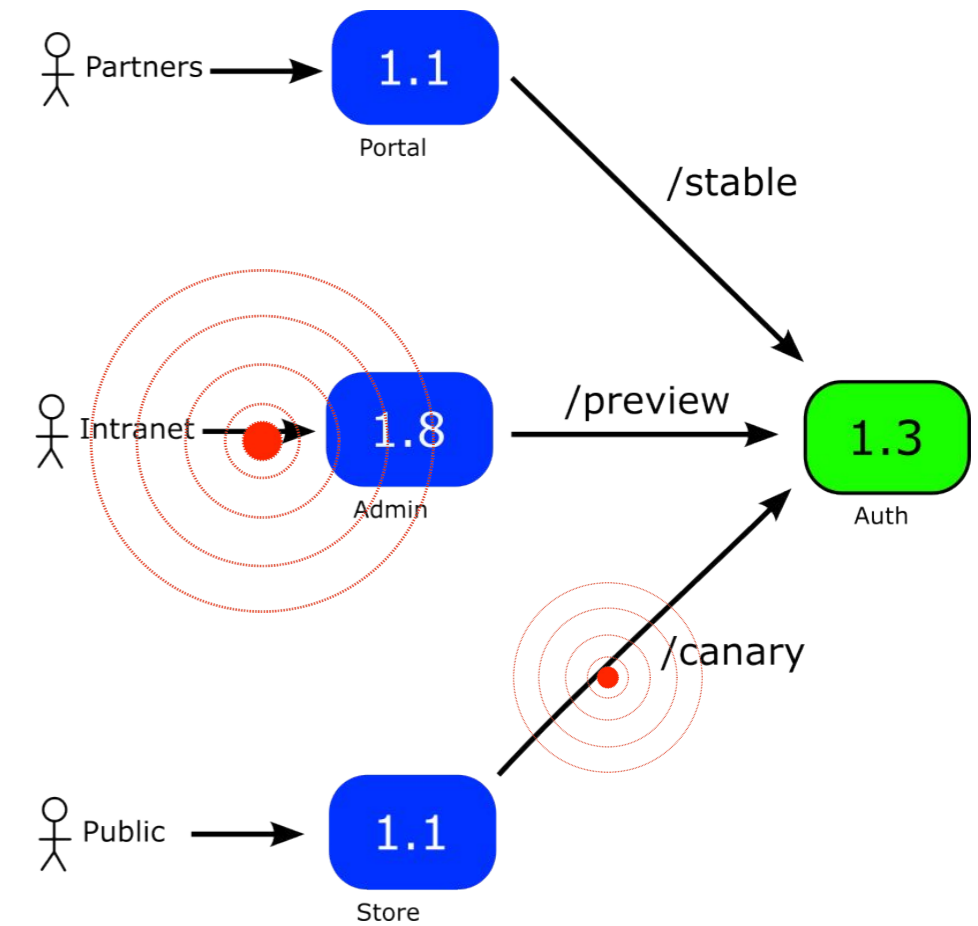


# Targeted Canaries

- Default canaries are request based
- Canaries affect random users
- Same blast radius for everybody

Approach 1 - static URL routing (simple)

Approach 2 - dynamic header routing (advanced)





# What about feature flags

You can use feature flags in tandem with canaries

Feature flags **NEED** source code changes

Feature flags need ongoing maintenance

Static/dynamic routing is a one-off investment

Feature flags help with releases but not with deployments

 Open  
Feature



# Comparison

	Static URL Routing	Dynamic Header routing	Feature flags
Control your blast radius	Yes ✓ (deployment)	Yes ✓ (deployment)	Yes ✓ (release)
Requires source code changes	No ✓	Maybe ✗ ✓	YES ✗
Adoption effort	Low ✓	Medium ✗ ✓	High ✗
Single time setup	Yes ✓	Yes ✓	No ✗
Fully dynamic	No ✗	Yes ✓	Yes ✓
Canaries targeted at users/user groups	No ✗	Yes ✓	Yes ✓





# Thank you!

Questions: [kostis.kapelonis@octopus.com](mailto:kostis.kapelonis@octopus.com)

GitOps/Argo CD certification [learning.codefresh.io](https://learning.codefresh.io)

CNCF Slack <https://slack.cncf.io/>

Rollouts <https://argoproj.github.io/rollouts/>

 Octopus Deploy

# Backup Slides

