

Kubernetes - Part 6

Solar Team

Kubernetes Deployment Strategies

- RollingUpdate Deployment
- Recreate Deployment
- Advance Strategies
 - Blue/green deployment
 - Canary deployment
 - By Label Selectors
 - By Header (using Nginx Controller)
 - By Weight (using Nginx Controller)

RollingUpdate Deployment

- The Deployment updates Pods in a **rolling update fashion**.
- RollingUpdate Deployments support running multiple versions of an application at the same time.
- Specify **maxUnavailable** and **maxSurge value** to control the rolling update process.
- The value can be an **absolute number** (for example, 5) or a **percentage of desired Pods**.
- **MaxSurge** indicates how many extra pods(over the desired number) we are willing to run during a rolling update.
- **MaxUnavailable** indicates how many pods we can lose during the rolling update.
- Both parameters can be zero (but not at the same time), the default value is 25%.
- **progressDeadlineSeconds** - the maximum time in seconds(defaults to 600) for a deployment to make progress before it is considered to be failed.
- **revisionHistoryLimit** specifies the number(default to 10) of **old ReplicaSets** to retain to allow rollback.

Updating a Deployment

- A Deployment's rollout is triggered if and only if the Deployment's Pod template (that is, `.spec.template`) is **changed**, for example if the labels or container images of the template are updated.
- Other updates, such as scaling the Deployment, do not trigger a rollout.
- Update your Deployment by:
 - `kubectl apply`
 - `kubectl set`
 - `kubectl edit`
 - `kubectl patch`
- Checking the rollout status
 - `kubectl rollout status`
 - `kubectl get deployments`
 - `kubectl get rs`
 - `kubectl get pods`
- **Rollover** (aka multiple updates in-flight)
- **Pod-template-hash** label (**Caution: Do not change this label.**)

Updating a Deployment - Example

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hello-app
  namespace: default
spec:
  replicas: 3
  selector:
    matchLabels:
      app: hello-app
  template:
    metadata:
      labels:
        app: hello-app
    spec:
      containers:
        - image:
gcr.io/google-samples/hello-app:1.0
          imagePullPolicy: IfNotPresent
          name: hello-app
          resources: {}
          restartPolicy: Always
```

hello-app-deploy.yml

- `kubectl apply -f hello-app-deploy.yml`
- `kubectl rollout status deploy hello-app`
- `kubectl rollout history deploy hello-app`
- `kubectl get deploy hello-app -oyaml > apply-v1.yml`
- `kubectl apply -f hello-app-deploy-chg-image.yml --record`
- `kubectl get deploy hello-app -oyaml > apply-v2.yml`
- **OR** `kubectl set image deploy hello-app hello-app=gcr.io/google-samples/hello-app:2.0 --record`
- `kubectl rollout history deploy hello-app`

Rolling Update - History, Undo, Pause and Resume

- Check the revisions of this Deployment
 - `kubectl rollout history <deployment>`
- To see the details of each revision
 - `kubectl rollout history <deployment> --revision=<revision number>`
- Rolling Back to a Previous Revision
 - `kubectl rollout undo <deployment>`
- Rollback to a specific revision
 - `kubectl rollout undo <deployment> --to-revision=<revision number>`
- Pause a Deployment before triggering one or more updates
 - `kubectl rollout pause <deployment>`
- Resume the Deployment
 - `kubectl rollout resume <deployment>`

Rolling Update

- `kubectl rollout history deploy hello-app`
- `kubectl rollout history deploy hello-app --revision 2`
- `kubectl rollout undo deploy hello-app`
- **OR** `kubectl rollout undo deploy hello-app [--to-revision ?]`
- `kubectl edit deploy hello-app` # don't keep a record
- `kubectl rollout history deploy hello-app`
- Wrong image version!
 - `kubectl set image deploy hello-app hello-app=gcr.io/google-samples/hello-app:3.0 --record`
- Pause/Resume example
 - `kubectl rollout pause deployment hello-app`
 - `kubectl set image deploy hello-app hello-app=gcr.io/google-samples/hello-app:1.0 --record` # what happen!
 - `kubectl rollout resume deployment hello-app`

Recreate Deployment

- All existing Pods are killed before new ones are created.
- `.spec.strategy.type==Recreate`
- Expect downtime.

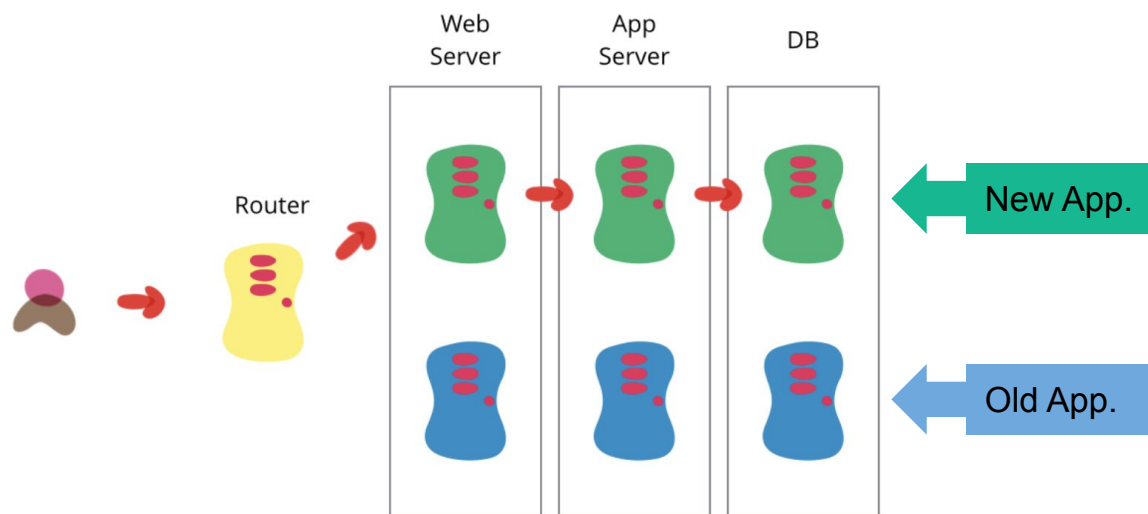
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hello-app-recreate
  namespace: default
spec:
  strategy:
    type: Recreate
  replicas: 3
  selector:
    matchLabels:
      app: hello-app-recreate
  template:
    metadata:
      labels:
        app: hello-app-recreate
    spec:
      containers:
      - image: gcr.io/google-samples/hello-app:1.0
        imagePullPolicy: IfNotPresent
        name: hello-app
        resources: {}
        restartPolicy: Always
```

hello-app-recreate.yml

- `kubectl apply -f hello-app-recreate.yml`
- `kubectl set image deploy hello-app-recreate hello-app=gcr.io/google-samples/hello-app:2.0 --record`

Blue/green Deployment

- Instantly switch over all the traffic from the old version to the new, instead of doing it **progressively**.
- Have enough hardware resources(on production) for deploying a new version of application.
- Blue-green deployment gives you a rapid way to rollback.
- Long-running transaction issues in the blue environment.



Credit: <https://martinfowler.com/>

Blue/green Deployment Example - Deploy Blue

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hello-app-v1
  namespace: default
spec:
  replicas: 3
  selector:
    matchLabels:
      app: hello-app
      version: v1.0
  template:
    metadata:
      labels:
        app: hello-app
        version: v1.0
    spec:
      containers:
      - image:
gcr.io/google-samples/hello-app:1.0
        imagePullPolicy: IfNotPresent
        name: hello-app
        resources: {}
        restartPolicy: Always
```

hello-v1-deploy.yml

```
apiVersion: v1
kind: Service
metadata:
  name: hello-bluegreen
spec:
  selector:
    app: hello-app
    version: v1.0
  ports:
    - protocol: TCP
      port: 8062
      targetPort: 8080
```

hello-bluegreen.yml

- `kubectl apply -f hello-v1-deploy.yml`
- `kubectl apply -f hello-bluegreen.yml`
- `curl hello-bluegreen:8062`

Blue/green Deployment Example - Deploy Green

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hello-app-v2
  namespace: default
spec:
  replicas: 3
  selector:
    matchLabels:
      app: hello-app
      version: v2.0
  template:
    metadata:
      labels:
        app: hello-app
        version: v2.0
    spec:
      containers:
      - image: gcr.io/google-samples/hello-app:2.0
        imagePullPolicy: IfNotPresent
        name: hello-app
        resources: {}
        restartPolicy: Always
```

hello-v2-deploy.yml

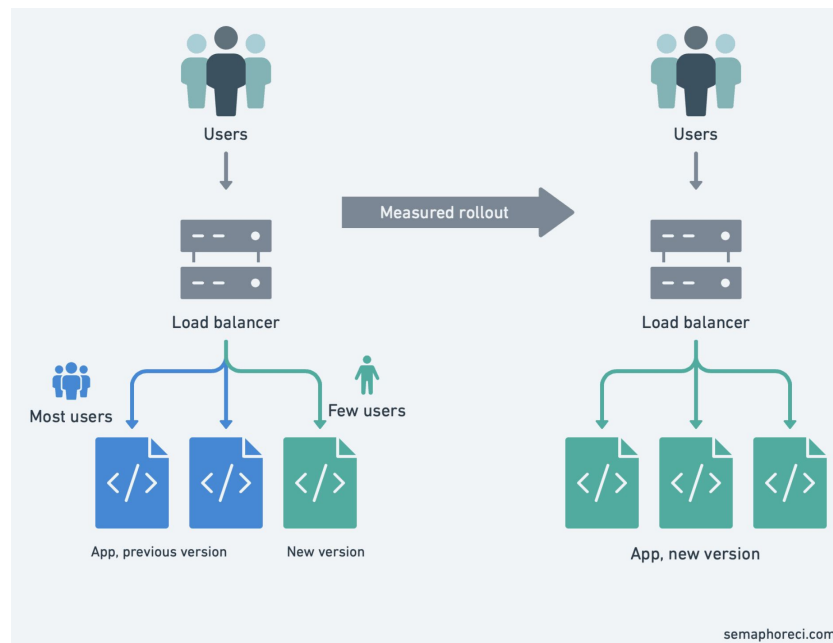
```
apiVersion: v1
kind: Service
metadata:
  name: hello-bluegreen
spec:
  selector:
    app: hello-app
    version: v1.0
  ports:
    - protocol: TCP
      port: 8062
      targetPort: 8080
```

hello-bluegreen.yml

- `kubectl apply -f hello-v2-deploy.yml`
- `switch to v2.0 pods`
 - `kubectl patch service hello-bluegreen -p '{"spec": {"selector": {"app": "hello-app", "version": "v2.0"}}}'`
- `curl hello-bluegreen:8062 # should be return v2.0`

Canary Deployment

- Canary deployment is a technique to reduce the risk of introducing a new software version in production.
- Allows you to test your new service version on a real small user group without a full rollout.
- Compare metrics between the current version and the canary that we just deployed.



Canary Deployment - By Label Selectors

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hello-app-v1
  namespace: default
spec:
  replicas: 3
  selector:
    matchLabels:
      app: hello-app
      version: v1.0
      enabled: "true"
  template:
    metadata:
      labels:
        app: hello-app
        version: v1.0
        enabled: "true"
    spec:
```

hello-v1-deploy.yml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hello-app-v2
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: hello-app
      version: v2.0
      enabled: "true"
  template:
    metadata:
      labels:
        app: hello-app
        version: v2.0
        enabled: "true"
    spec:
```

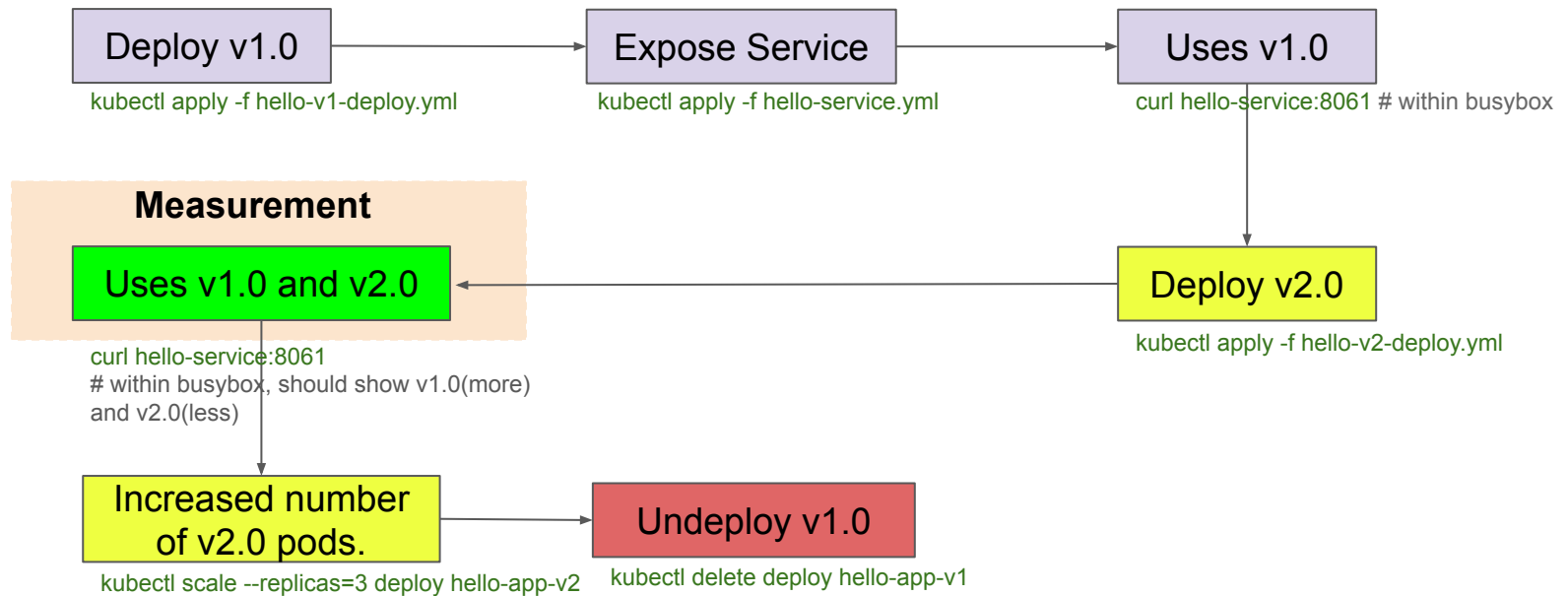
hello-v2-deploy.yml

```
apiVersion: v1
kind: Service
metadata:
  name: hello-service
spec:
  selector:
    app: hello-app
    enabled: "true"
  ports:
    - protocol: TCP
      port: 8061
      targetPort: 8080
```

hello-service..yml

- **kubectl apply -f hello-v1-deploy.yml**
- **kubectl apply -f hello-service.yml**
- **curl hello-service:8061 # within busybox**
- **kubectl apply -f hello-v2-deploy.yml**
- **curl hello-service:8061 # within busybox, should show v1.0(more) and v2.0(less)**
- **kubectl scale --replicas=3 deploy hello-app-v2**
- **curl hello-service:8061 # within busybox, v1.0 and v2.0 should be called at similar rates.**
- **kubectl delete deploy hello-app-v1**

Canary Deployment - By Label Selectors - Flows



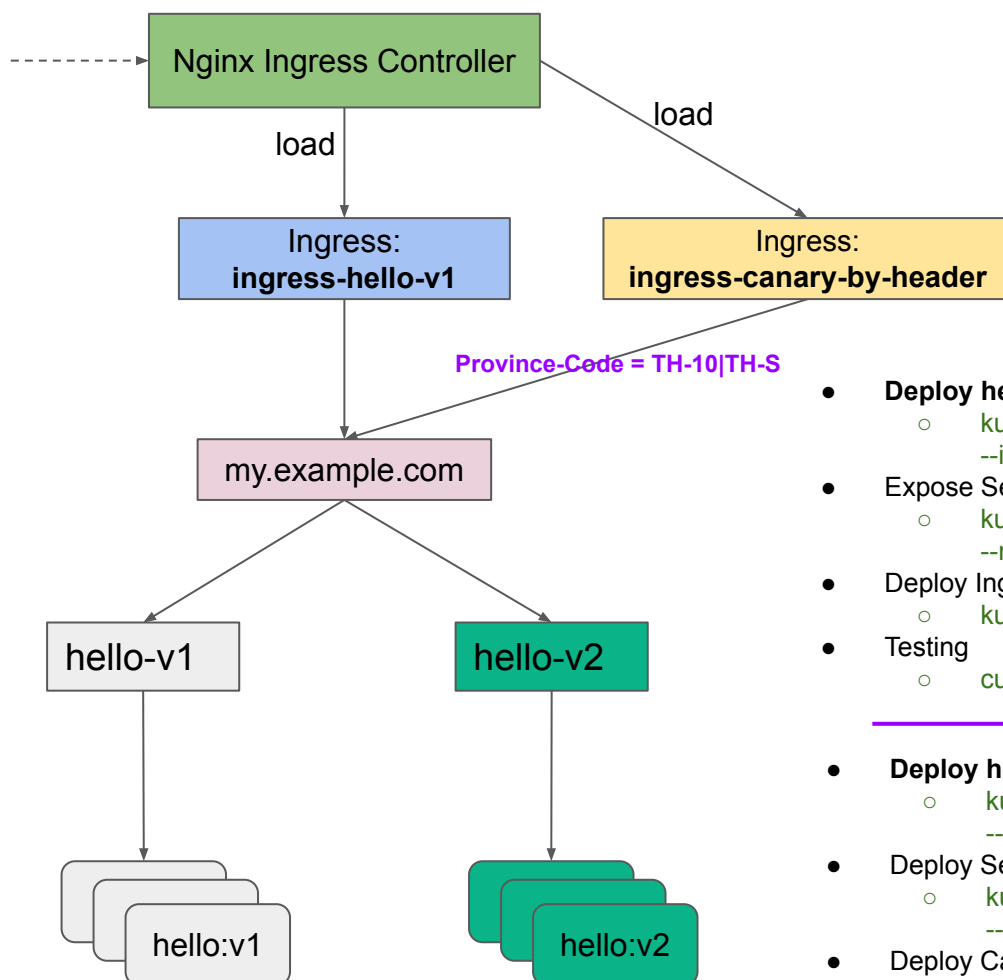
- If version 2 failed.
 - `kubectl delete deploy hello-app-v2`

Canary Deployment Example - By Header

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-canary-by-header
  annotations:
    kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/rewrite-target: /
    nginx.ingress.kubernetes.io/canary: "true"
    nginx.ingress.kubernetes.io/canary-by-header: "Province-Code"
    nginx.ingress.kubernetes.io/canary-by-header-pattern: "TH-10|TH-S"
spec:
  rules:
    - host: my.example.com
      http:
        paths:
          - path: /hello-test
            pathType: Prefix
            backend:
              service:
                name: hello-v2
                port:
                  number: 8060
```

ingress-canary-by-header.yml

Canary Deployment Example - By Header

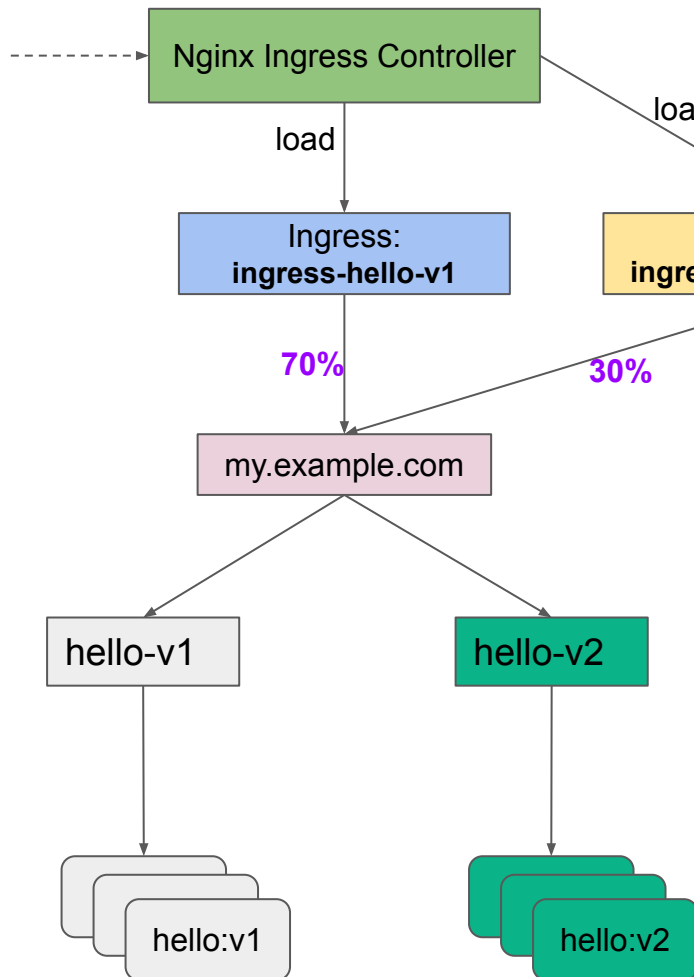


- **Deploy hello application version 1**
 - `kubectl create deployment hello-app-v1 --image=gcr.io/google-samples/hello-app:1.0`
 - **Expose Service version 1**
 - `kubectl expose deployment hello-app-v1 --port=8060 --target-port=8080 --name=hello-v1`
 - **Deploy Ingress resource**
 - `kubectl apply -f ingress-hello-v1.yml`
 - **Testing**
 - `curl http://my.example.com/hello-test`
-
- **Deploy hello application Version 2**
 - `kubectl create deployment hello-app-v2 --image=gcr.io/google-samples/hello-app:2.0`
 - **Deploy Service version 2**
 - `kubectl expose deployment hello-app-v2 --port=8060 --target-port=8080 --name=hello-v2`
 - **Deploy Canary Ingress resource**
 - `kubectl apply -f ingress-canary-by-header.yml`
 - **Testing**
 - `curl -H "Province-Code: TH-11" http://my.example.com/hello-test`

Canary Deployment - By Weight

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-canary-by-header
  annotations:
    kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/rewrite-target: /
    nginx.ingress.kubernetes.io/canary: "true"
    nginx.ingress.kubernetes.io/canary-weight: "30"
spec:
  rules:
  - host: my.example.com
    http:
      paths:
      - path: /hello-test
        pathType: Prefix
        backend:
          service:
            name: hello-v2
            port:
              number: 8060
```

Canary Deployment Example - by Weight



- **Deploy version 1**
 - `kubectl create deployment hello-app-v1 --image=gcr.io/google-samples/hello-app:1.0`
- **Expose Service**
 - `kubectl expose deployment hello-app-v1 --port=8060 --target-port=8080 --name=hello-v1`
- **Deploy Ingress resource**
 - `kubectl apply -f ingress-hello-v1.yml`
- **Testing**
 - `curl http://my.example.com/hello-test`

- **Deploy Version 2**
 - `kubectl create deployment hello-app-v2 --image=gcr.io/google-samples/hello-app:2.0`
- **Deploy Service for app version 2**
 - `kubectl expose deployment hello-app-v2 --port=8060 --target-port=8080 --name=hello-v2`
- **Deploy Canary Ingress resource**
 - `kubectl apply -f ingress-canary-by-weight.yml`
- **Testing**
 - `curl http://my.example.com/hello-test`

DEPLOYMENT STRATEGIES

When it comes to production, a ramped or blue/green deployment is usually a good fit, but proper testing of the new platform is necessary.

Blue/green and shadow strategies have more impact on the budget as it requires double resource capacity. If the application lacks in tests or if there is little confidence about the impact/stability of the software, then a canary, a/b testing or shadow release can be used.

If your business requires testing of a new feature amongst a specific pool of users that can be filtered depending on some parameters like geolocation, language, operating system or browser features, then you may want to use the a/b testing technique.



Strategy	ZERO DOWNTIME	REAL TRAFFIC TESTING	TARGETED USERS	CLOUD COST	ROLLBACK DURATION	NEGATIVE IMPACT ON USER	COMPLEXITY OF SETUP
RECREATE version A is terminated then version B is rolled out	✗	✗	✗	■ □ □	■ ■ ■	■ ■ ■	□ □ □
RAMPED version B is slowly rolled out and replacing version A	✓	✗	✗	■ □ □	■ ■ ■	■ □ □	■ □ □
BLUE/GREEN version B is released alongside version A, then the traffic is switched to version B	✓	✗	✗	■ ■ ■	□ □ □	■ ■ □	■ ■ □
CANARY version B is released to a subset of users, then proceed to a full rollout	✓	✓	✗	■ □ □	■ □ □	■ □ □	■ ■ □
A/B TESTING version B is released to a subset of users under specific condition	✓	✓	✓	■ □ □	■ □ □	■ □ □	■ ■ ■
SHADOW version B receives real world traffic alongside version A and doesn't impact the response	✓	✓	✗	■ ■ ■	□ □ □	□ □ □	■ ■ ■

Be careful when using label and selector

```
k (kubectl)
Hostname: hello-app-f99cf888-4v62x
/ # curl hello
curl: (6) Couldn't resolve host 'hello'
/ # curl hello
Hello, world!
Version: 1.0.0
Hostname: hello-app-f99cf888-lv8d8
/ # curl hello
curl: (6) Couldn't resolve host 'hello'
/ # curl hello
curl: (6) Couldn't resolve host 'hello'
/ # curl hello
Hello, world!
Version: 1.0.0
Hostname: hello-app-f99cf888-781lqq
/ # curl hello
curl: (6) Couldn't resolve host 'hello'
/ # curl hello
curl: (6) Couldn't resolve host 'hello'
/ # curl hello
Hello, world!
Version: 1.0.0
Hostname: hello-app-f99cf888-781lqq
/ # curl hello
curl: (7) Failed to connect to hello port 80: Connection refused
/ # curl hello
curl: (6) Couldn't resolve host 'hello'
/ # curl hello
Hello, world!
Version: 1.0.0
Hostname: hello-app-f99cf888-jqwkv
/ # curl hello
curl: (6) Couldn't resolve host 'hello'
/ # curl hello
Hello, world!
Version: 1.0.0
Hostname: hello-app-f99cf888-781lqq
/ # :

~ (zsh)
NAME          ENDPOINTS          AGE
echo          10.1.2.142:8080    2d17h
hello        10.1.2.198:8080,10.1.2.199:8080,10.1.2.200:8080 + 9 more... 2d17h
kubernetes    192.168.65.3:6443  177d
nginx        10.1.2.139:80,10.1.2.141:80,10.1.2.144:80 + 3 more... 2d22h
nodeport-service 10.1.2.139:80,10.1.2.141:80,10.1.2.144:80 + 3 more... 2d17h

13:03:39 ~
~ (zsh)
Name: hello
Namespace: default
Labels: app=hello-app
Annotations: endpoints.kubernetes.io/last-change-trigger-time: 2021-06-11T05:59:57Z
Subsets:
  Addresses: 10.1.2.198,10.1.2.199,10.1.2.200,10.1.2.201,10.1.2.202,10.1.2.203,10.1.2.204,10.1.2.205,10.1.2.206,10.1.2.207,10.1.2.209,10.1.2.213
  NotReadyAddresses: <none>
  Ports:
    Name      Port      Protocol
    ----      -
    <unset> 8080     TCP

Events: <none>

13:03:49 ~
~ (zsh)

Context: docker-desktop
Cluster: docker-desktop
User: docker-desktop
K9s Rev: v0.24.2
K8s Rev: v1.14.7

<0> all
<1> test
<2> ingress-nginx
<3> default

<a> At...
<ctrl-d> De...
<d> De...
<e> Ed...
<?> He...
<ctrl-k> Kl...

-1 app=hello-app

Pod(s)(default)[12] </app=hello-app>
NAME          PF  READY  RESTARTS  STATUS  IP          NODE          AGE
hello-app-f99cf888-4v62x  ●  1/1    0          Running 10.1.2.203  docker-desktop 20h
hello-app-f99cf888-4zqgt  ●  1/1    0          Running 10.1.2.199  docker-desktop 20h
hello-app-f99cf888-556zk  ●  1/1    0          Running 10.1.2.207  docker-desktop 20h
hello-app-f99cf888-781lqq ●  1/1    0          Running 10.1.2.200  docker-desktop 20h
hello-app-f99cf888-9vw89  ●  1/1    0          Running 10.1.2.206  docker-desktop 20h
hello-app-f99cf888-jqwkv  ●  1/1    0          Running 10.1.2.205  docker-desktop 20h
hello-app-f99cf888-lv8d8  ●  1/1    0          Running 10.1.2.198  docker-desktop 20h
hello-app-f99cf888-vhctn  ●  1/1    0          Running 10.1.2.204  docker-desktop 20h
hello-app-f99cf888-vpb4d  ●  1/1    0          Running 10.1.2.202  docker-desktop 20h
hello-app-f99cf888-znong  ●  1/1    0          Running 10.1.2.201  docker-desktop 20h
nginx-fake     ●  1/1    0          Running 10.1.2.213  docker-desktop 16m
nginx-new-69dc468f6f-659cd ●  1/1    0          Running 10.1.2.209  docker-desktop 17h

<pod>

Context: docker-desktop
Cluster: docker-desktop
User: docker-desktop
K9s Rev: v0.24.2
K8s Rev: v1.14.7

<0> all
<1> default

<ctrl-l> Bench Ru...
<ctrl-d> Delete
<d> Describe
<e> Edit
<?> Help
<l> Logs

NAME          TYPE          CLUSTER-IP  EXTERNAL-IP  PORTS          AGE
echo          ClusterIP    10.97.83.180  <unset>      80->0          2d17h
hello        ClusterIP    10.102.9.13  <unset>      80->0          2d17h
kubernetes    ClusterIP    10.96.0.1    https:443->0 177d
nginx        NodePort     10.108.145.58 8080->80:8080->31013 2d22h
nodeport-service NodePort     10.101.175.44 8080->30238 2d17h
other-ns-service ExternalName  <unset>      <unset>      hello.test.svc.cluster.local 43h

<service>
```

End.