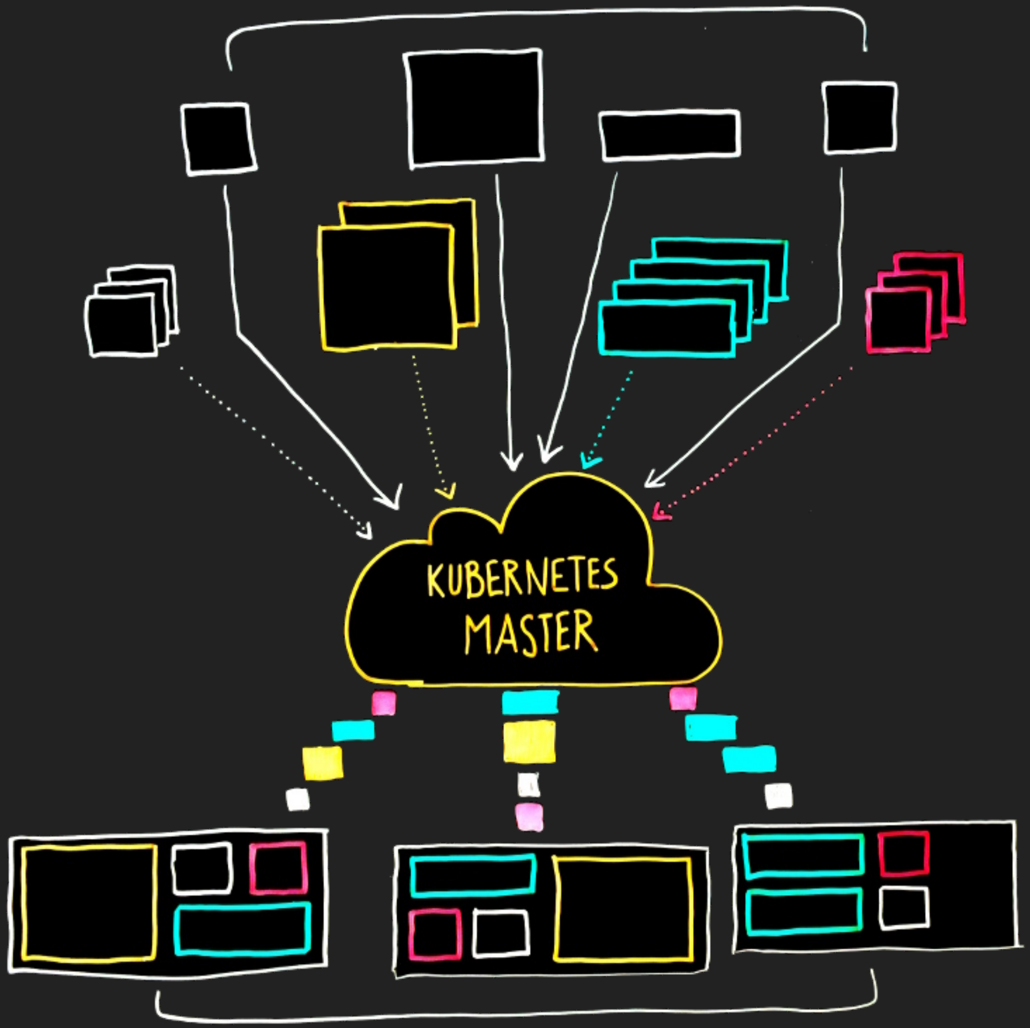
CD and Kubernetes - day after

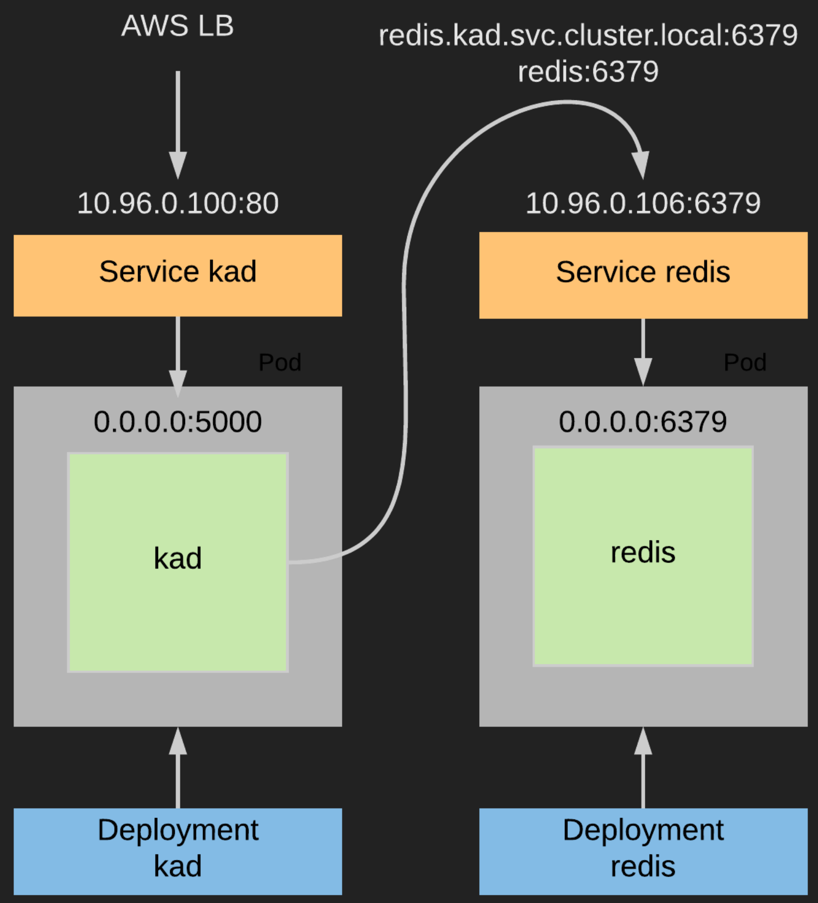
prgcont.cz

Tomáš Kukrál

2019-11-13


KUBERNETES
MASTER

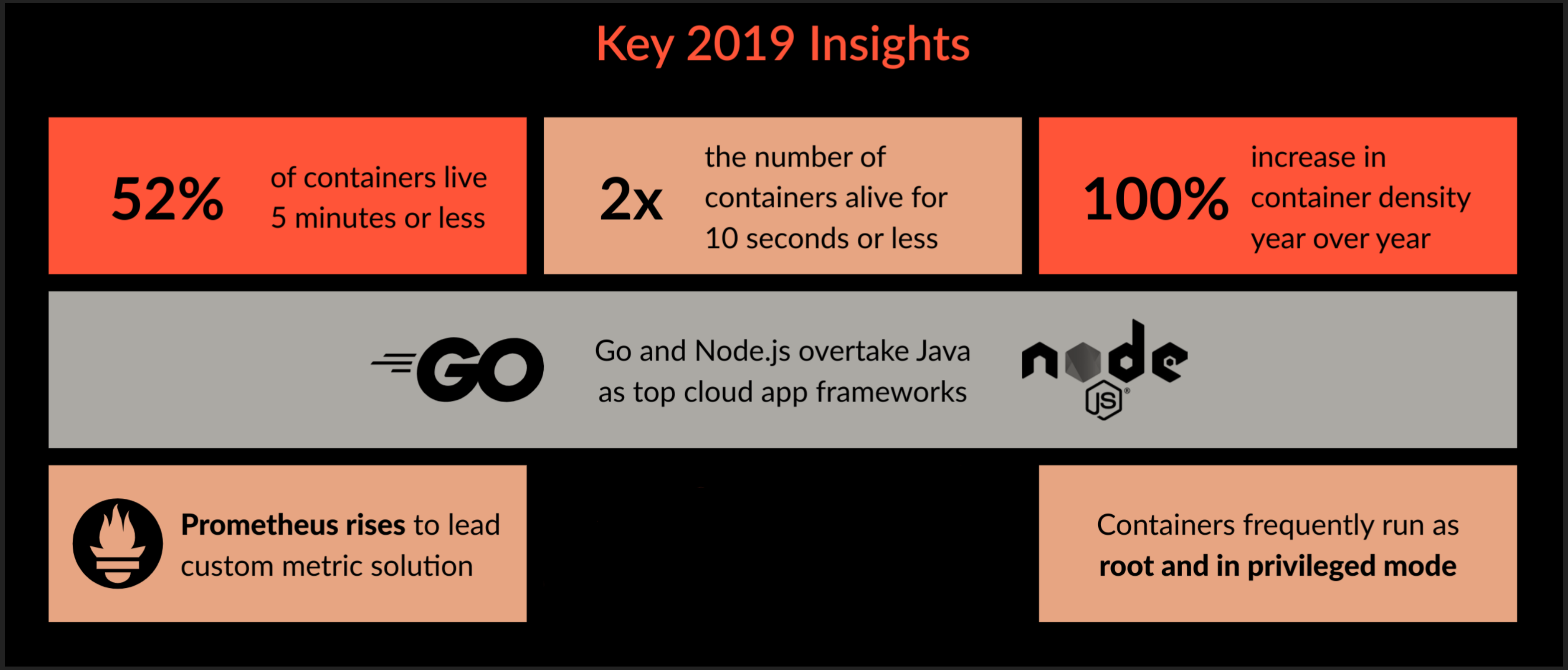
Day 1



- Kubernetes cluster
- Deployments
- Services
- LoadBalancer
- ConfigMaps

- Public DNS records
- TLS certificates
- Backups

Day 2



Key 2019 Insights

52% of containers live 5 minutes or less

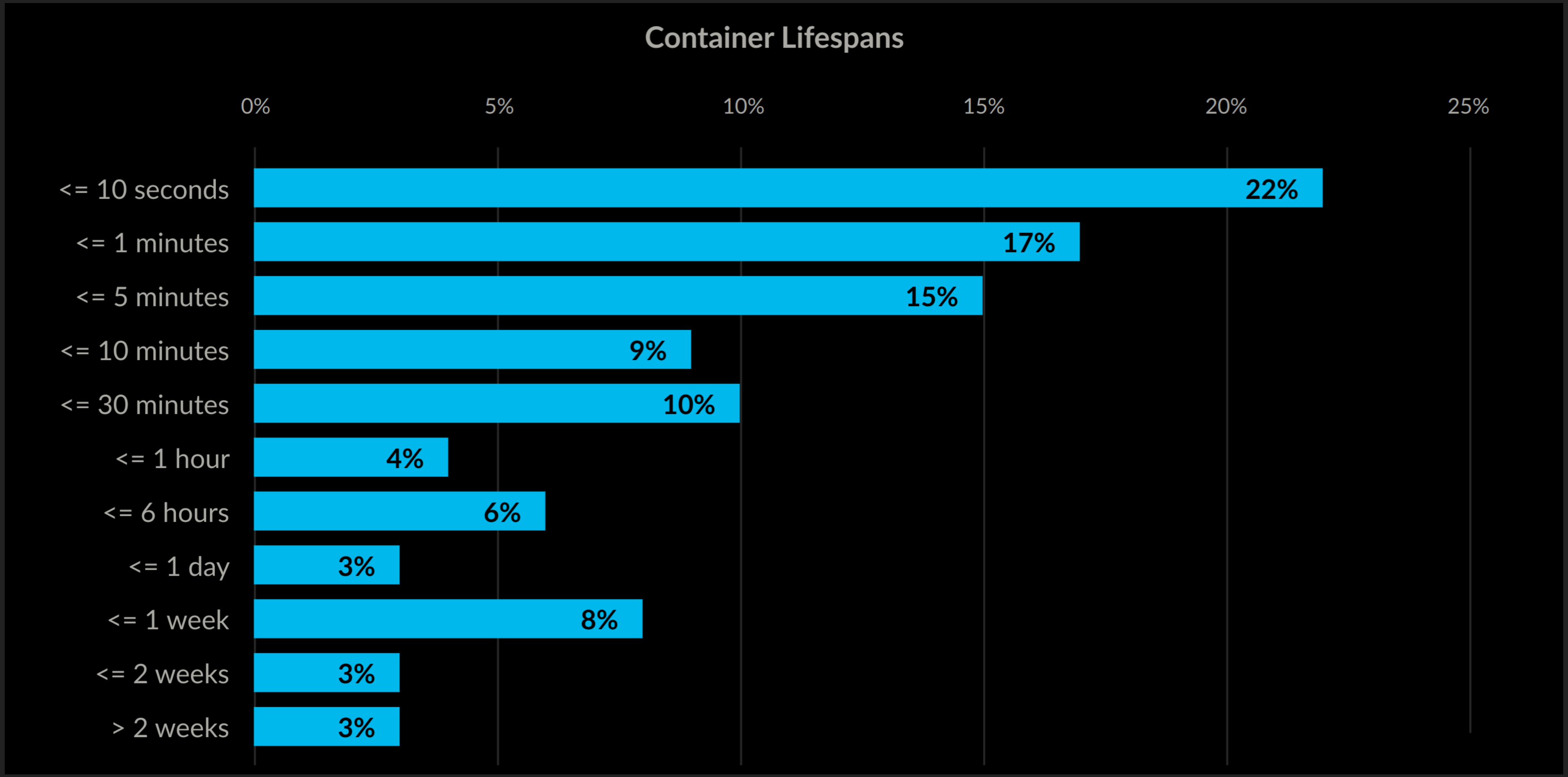
2x the number of containers alive for 10 seconds or less

100% increase in container density year over year

Go and Node.js overtake Java as top cloud app frameworks

Prometheus rises to lead custom metric solution

Containers frequently run as **root and in privileged mode**



Container Lifespans
0%
5%
10%
15%
20%
25%
<= 10 seconds
22%
<= 1 minutes
17%
<= 5 minutes
15%
<= 10 minutes
9%
<= 30 minutes
10%
<= 1 hour
4%
<= 6 hours
6%
<= 1 day
3%
<= 1 week
8%
<= 2 weeks
3%
> 2 weeks
3%

Run CD pipeline

Declaring declarative objects

```yaml
---
apiVersion: v1
kind: Service
metadata:
  name: kad
spec:
  type: LoadBalancer
  selector:
    app: kad
  ports:
  - protocol: TCP
    port: 80
    targetPort: 5000
status:
  loadBalancer:
    ingress:
    - hostname: a13.eu-central-1.elb.amazonaws.com
```

Update version (image)

```
spec:
  containers:
  - name: app
     image: tomkukral/kad:v1.4.5
```

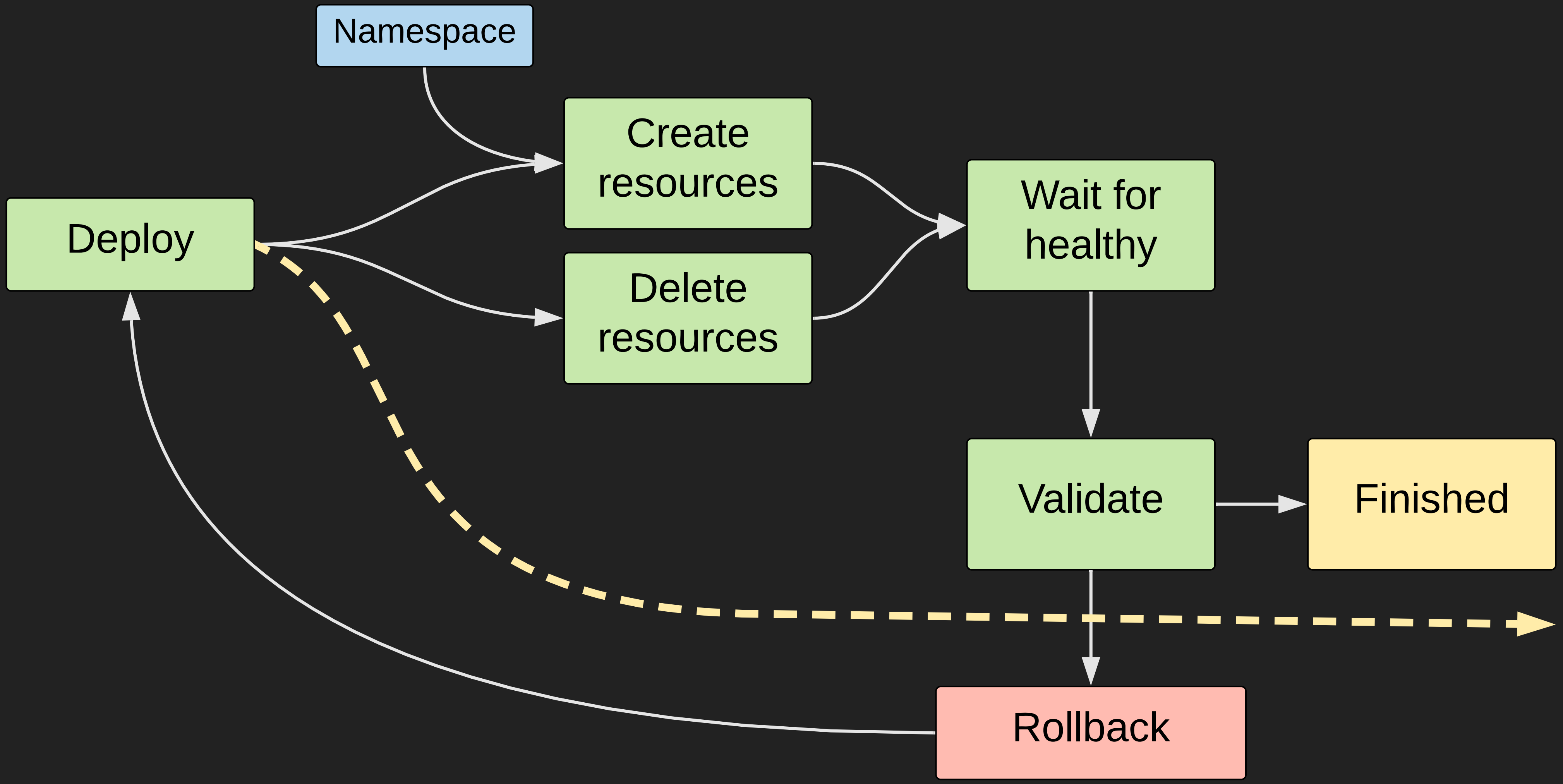
```
spec:
  containers:
  - name: app
     image: tomkukral/kad:v2.0.1
```

Change immutable fields

```
apiVersion: v1
kind: Service
metadata:
  name: kad
spec:
  type: NodePort
```

```
apiVersion: v1
kind: Service
metadata:
  name: kad
spec:
  type: ClusterIP
```

Delete resources?


Namespace
Deploy
Create resources
Delete resources
Wait for healthy
Validate
Finished
Rollback

Merge strategies

```
Ports []ServicePort `json:"ports,omitempty" patchStrategy:"merge" patchMergeKey:"port" ...`
```

- ServiceSpec

Maintenance operations

Deployment process options parameters

- Optimistic deployment
- Pre-update actions
- Retries

- Immutable fields

- Multiple environments
    - Canary, A/B, brown/purple

Kubernetes SDK

- **Python**
- **Java**
- **client-go**

```
go get k8s.io/client-go@kubernetes-1.15.3
```

```
config, err := clientcmd.BuildConfigFromFlags("", "/home/tom/.kube/config")
clientset, err := kubernetes.NewForConfig(config)
```

Everything can fail

```go
// Retry provide retry with default parameters
func Retry(rf retry.RetryableFunc) error {
    return retry.Do(
        rf,
        retry.Attempts(300),
        retry.OnRetry(func(n uint, err error) {
            log.Printf("#%d: %s\n", n, err)
        }),
        retry.DelayType(func(uint, *retry.Config) time.Duration { return 400 * time.Millisecond }),
    )
}


if err := sdk.Retry(func () error {
    return checkEndpoints(cs, svc.ObjectMeta.Name)
}); err != nil {
    panic(err)
}
```

Check manifest validity

```go
if !strings.Contains(v, "apiVersion: ") || !strings.Contains(v, "kind: ") {
    return fmt.Errorf("Manifest '%s' of app %s isn't valid, skipping", v, appName)
}
```

Add recommended labels

```go
an := rs.Obj.GetAnnotations()
an["app.kubernetes.io/managed-by"] = "vpm"
an["app.kubernetes.io/version"] = resp.Version
an["app.kubernetes.io/name"] = appName
```

Recommended labels - Kubernetes

Deployment strategy

```
// detect deployment strategy
if do, ok := an["ves.io/deploy"]; ok && do != "" {
  rs.DeployStrategy = do
}
```

- `optimistic` create resource and don't wait for status
- `pesimistic` wait for state

Prepull actions

```
// detect images to prepull
if v, ok := an["ves.io/prepull"]; ok && v != "" {
    for _, im := range strings.Split(v, "\n") {
        if im != "" {
      ... pull images on all nodes ...
        }
    }
}
```

- Pull images before changing pod resources

Give resources time to start

```
// wait for objects to be ready
if !k8s.WaitUntilObjectsRunning(ctx, waitObjs, m.ResourceWaitTimeout) {
    log.Printf("Failed waiting for Kubernetes objects, will check status for all of them")
}
```

QA