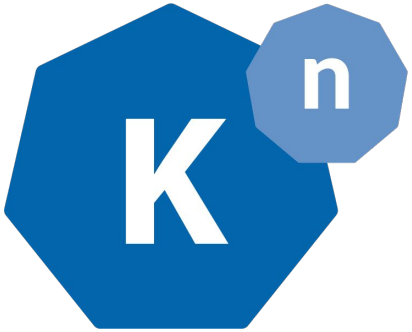


Extending Knative for fun and profit

Matt Moore @mattomata
Ville Aikas @AikasVille

Background

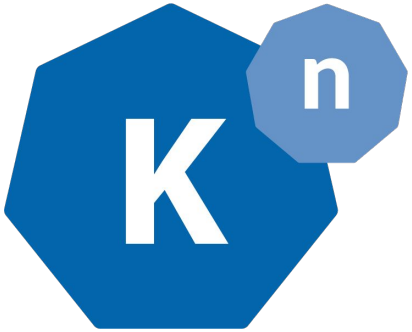
```
objectRef:
  apiVersion: serving.knative.dev/v1alpha1
  kind: Service
  name: vile
```

```
objectRef:
  apiVersion: my.corp.io/v3
  kind: Agent
  name: hutchinson
```

… but how to we deal with what's on the other side?

Option A: Bake it in!

```
package eventing

import (

    "github.com/knative/serving/..."
    "github.com/wesley/hutchinson/..."
    "github.com/colonel/mustard/..."
    "github.com/scarlet/speedster/..."
    "github.com/and/on/..."
    "github.com/and/on-forever/..."
    "github.com/please/no-more/..."
    "github.com/why/would/you/do/this/..."
    "github.com/omg/stop-it/..."
)
```



Option B) What the duck?

~~Extending Knative for fun and profit~~

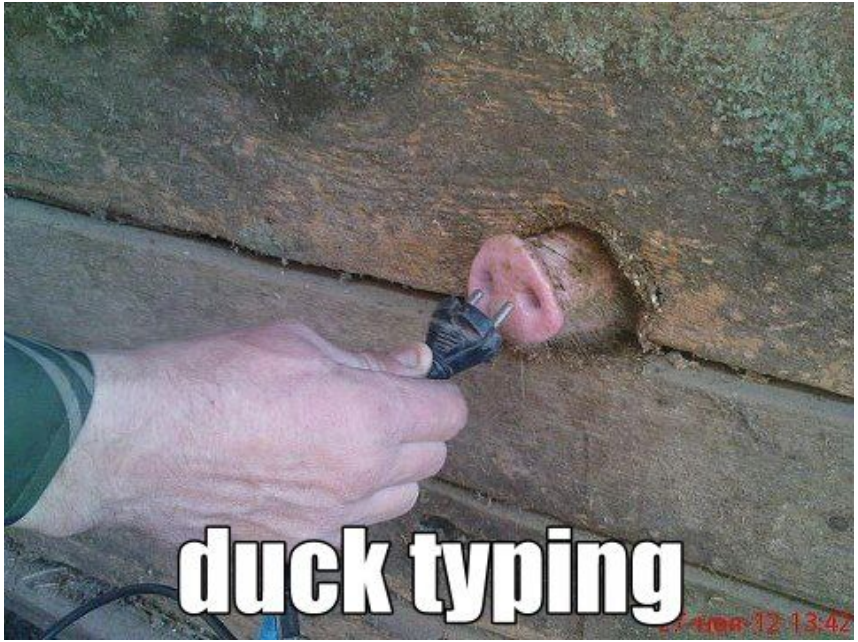
Matt Moore @mattomata
Ville Aikas @AikasVille

Scott Nichols @n3wscott (our mascot)

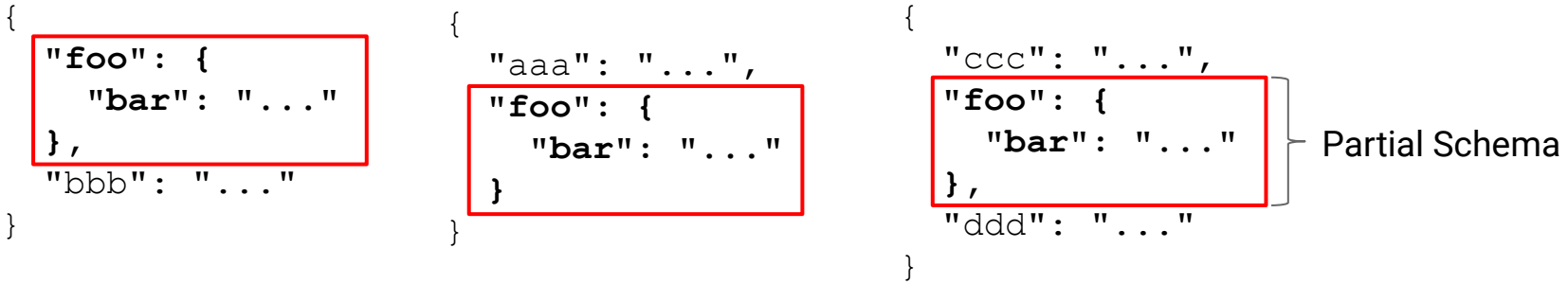
Duck Typing

*Duck typing in computer programming is an application of the duck test—"**If it walks like a duck and it quacks like a duck, then it must be a duck**"...*

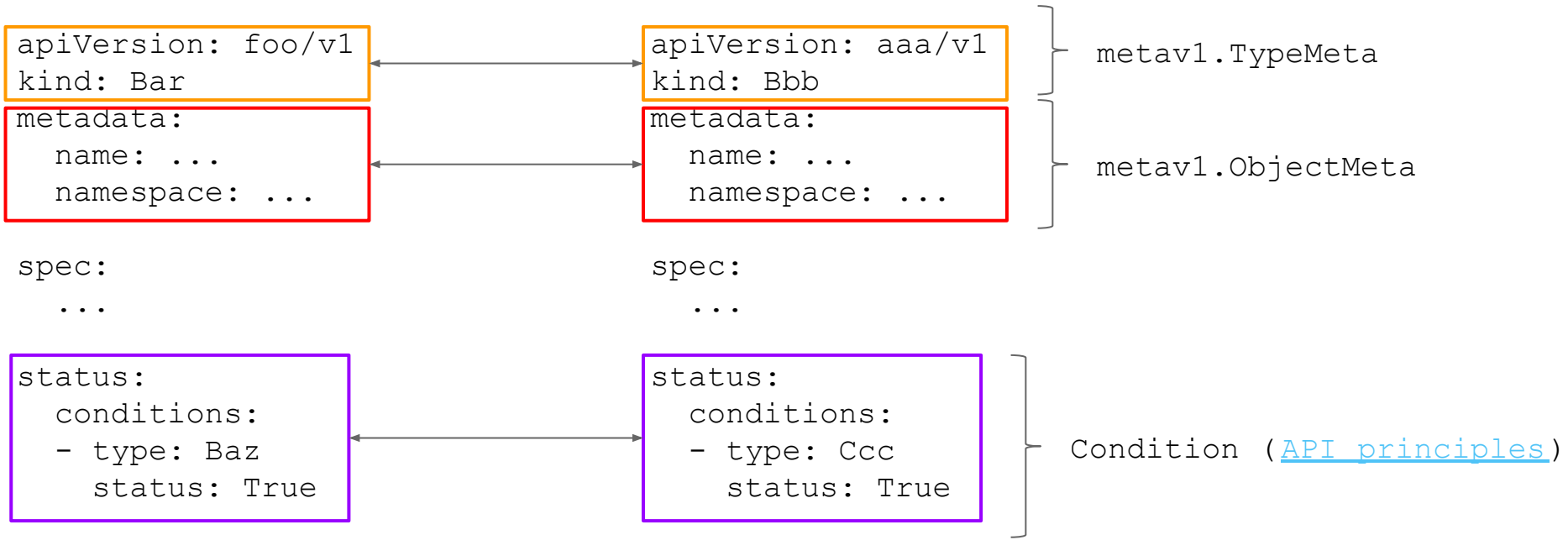
-- Wikipedia



JSON/YAML “duck typing” basics



Kubernetes Duck Types



Kubernetes Duck Types (cont'd)

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 3
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
```

(source)

```yaml
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: fluentd-elasticsearch
  labels:
    k8s-app: fluentd-logging
spec:
  selector:
    matchLabels:
      name: fluentd-elasticsearch
  template:
    metadata:
      labels:
        name: fluentd-elasticsearch
    spec:
      containers:
      - name: fluentd-elasticsearch
        image: fluentd:v2.5.1
```

(abbreviated source)

```yaml
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: web
spec:
  selector:
    matchLabels:
      app: nginx
  serviceName: "nginx"
  replicas: 3
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:0.8
```

(abbreviated source)

Formalizing the standard “Pod Spec” shape

```
type WithPod struct {
        metav1.TypeMeta   `json:",inline"`
        metav1.ObjectMeta `json:"metadata,omitempty"`

        Spec WithPodSpec `json:"spec,omitempty"`
}

type WithPodSpec struct {
        Template corev1.PodTemplateSpec `json:"template,omitempty"`
}
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 3
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
```

Demo #1 - “cachier”

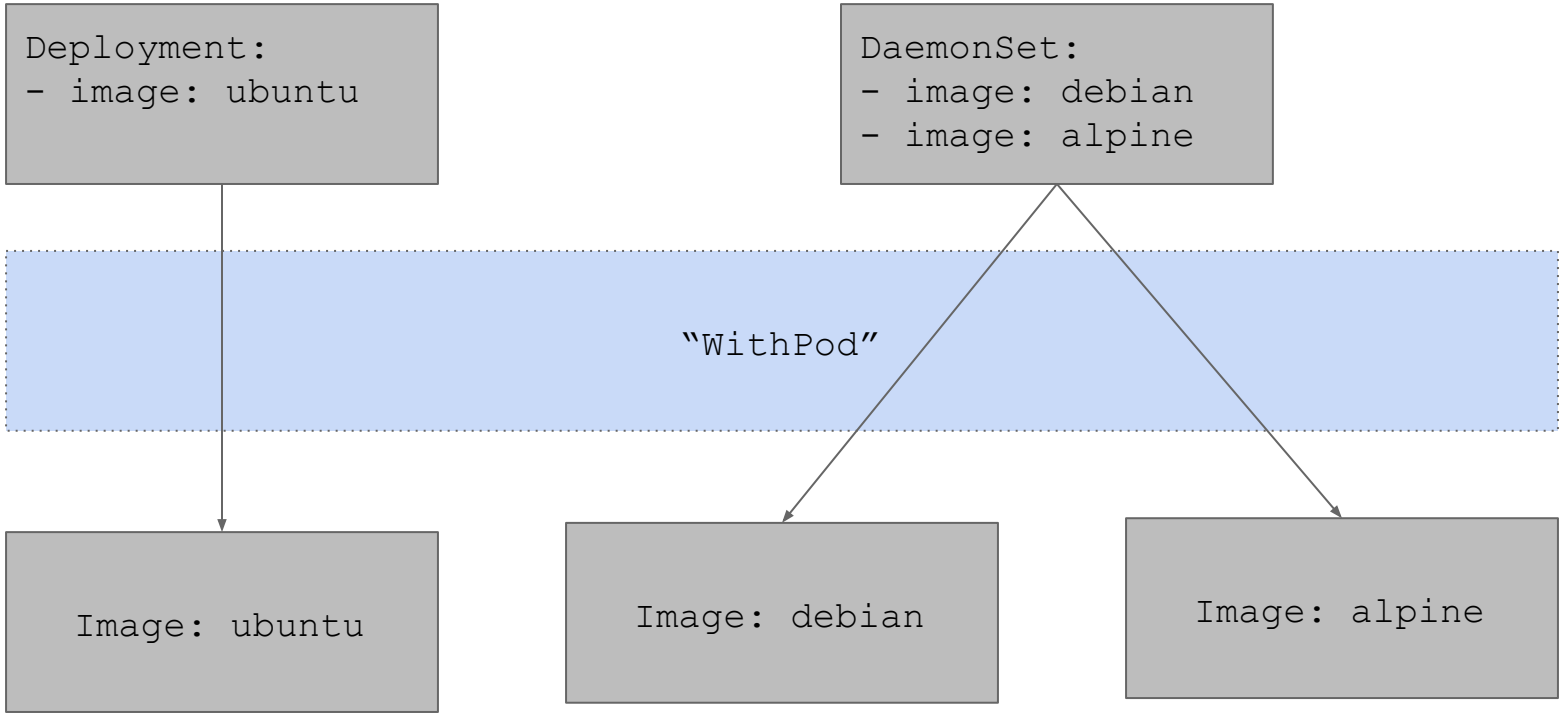
```go
type WithPod struct {
	metav1.TypeMeta   `json:",inline"`
	metav1.ObjectMeta `json:"metadata,omitempty"`

	Spec WithPodSpec `json:"spec,omitempty"`
}

type WithPodSpec struct {
	Template corev1.PodTemplateSpec `json:"template,omitempty"`
}
```

Reads the PodSpec from any K8s app resource, and creates “Image” resources.

```
14      spec:
15        serviceAccountName: cachier-controller
16        containers:
17        - name: cachier-controller
18          image: github.com/mattmoor/cachier/cmd/controller
19          args:
20          - "-logtostderr=true"
21          - "-stderrthreshold=INFO"
22          # Add PodSpecable types here:
23          - "-resource=Deployment.v1.apps"
24          - "-resource=ReplicaSet.v1.apps"
25          - "-resource=StatefulSet.v1.apps"
26          - "-resource=DaemonSet.v1.apps"
```

The extent of our awareness of these types

Demo #1 - “cachier”



Demo #1 - “cachier”

Let’s see it in action...

Recap: Demo #1 - “cachier”

We can use our partial schema to read data out of arbitrary resources, in this case enabling us to cache container images for arbitrary `WithPod` resources.

Note that like Go interfaces (also duck types), we were able to do this without the “app” resources having any knowledge of our partial schema, and for which the partial schema may not have been defined at the time they were authored.

… but what I we want to ***change*** arbitrary resources?

What if we want to UPDATE something?

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  sele
    hLabels
      app:
  rep
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
        env:
        - name: FOO
          value: BAR
```

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: fluentd-elasticsearch
  namespace: kube-system
  labels:
    k8s-app: fluentd-logging
spec:
  se
    mat
      name:
  template:
    metadata:
      labels:
        name: fluentd-elasticsearch
    spec:
      containers:
      - name: fluentd-elasticsearch
        image: fluentd:v2.5.1
        env:
        - name: FOO
          value: BAR
```

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: web
spec:
  sele
    hLabels:
      app
  repl
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:0.8
        env:
        - name: FOO
          value: BAR
```

Option A: Bake it in!

```
package eventing

import (

    "github.com/knative/serving/..."
    "github.com/wesley/hutchinson/..."
    "github.com/colonel/mustard/..."
    "github.com/scarlet/speedster/..."
    "github.com/and/on/..."
    "github.com/and/on-forever/..."
    "github.com/please/no-more/..."
    "github.com/why/would/you/do/this/..."
    "github.com/omg/stop-it/..."
)
```

What would the PATCH look like?

```
spec:
  template:
    spec:
      containers:
      - env:
        - name: FOO
          value: BAR
```

```
spec:
  template:
    spec:
      containers:
      - env:
        - name: FOO
          value: BAR
```

```
spec:
  template:
    spec:
      containers:
      - env:
        - name: FOO
          value: BAR
```

```
spec:
  template:
    spec:
      containers:
      - env:
        - name: FOO
          value: BAR
```

What it looks like against `WithPod`

(Merge patch shown)

The Patching corollary

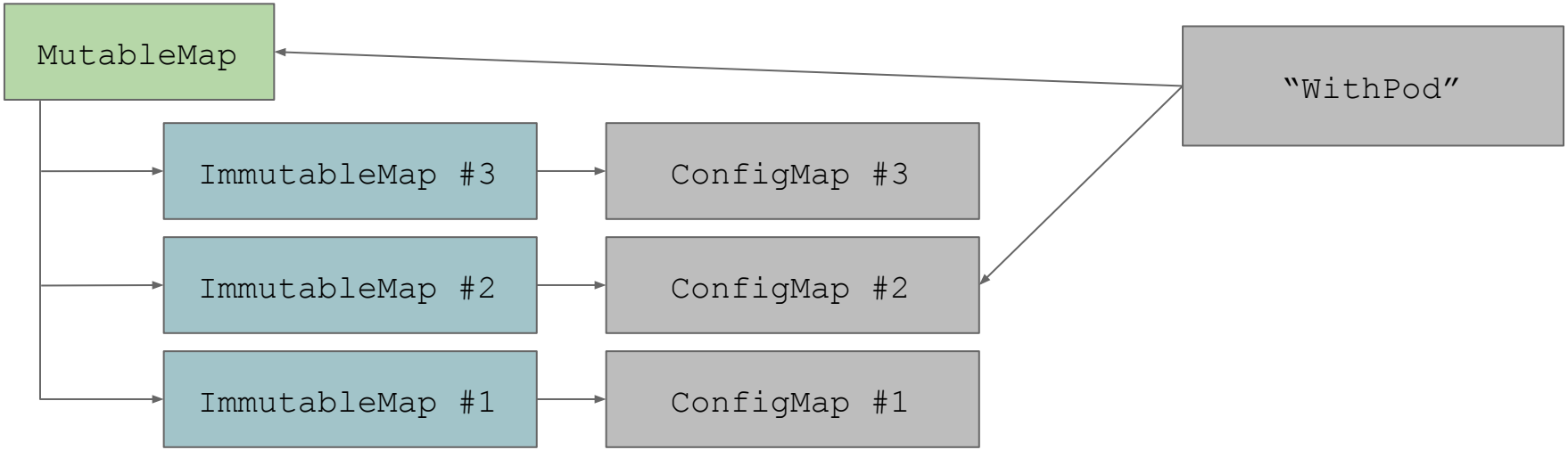
If we generate a PATCH on our partial schema (our 🦆 type), it updates just the right things.

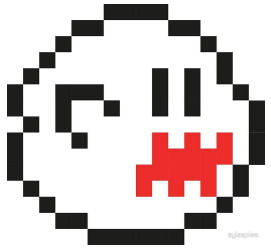
e.g. consider this patch:

```
[]jsonpatch.JsonPatchOperation{{
    Operation: "add",
    Path:      "/spec/template/spec/containers/0/env",
    Value:     []corev1.EnvVar{{Name: "FOO", Value: "bar"}},
}}
```

… applies equally well to all of the K8s app types.

Demo #2 - “boo-maps”





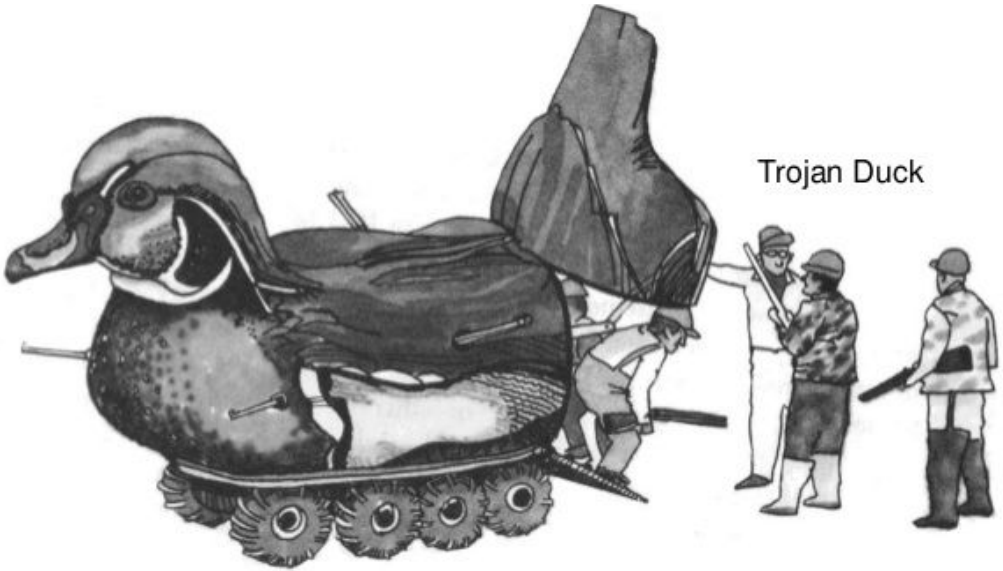
Demo #2 - “boo-maps”

Let’s see it in action...

Recap: Demo #2 - “boo-maps”

With this, the rollout of ConfigMap changes can be completely controlled through updates to the Deployment.

So what does all of this have to do with Knative, anyways?



Where it all began: Eventing -> Serving

```yaml
apiVersion: sources.eventing.knative.dev/v1alpha1
kind: GitHubSource
metadata:
  name: githubsourcesample
spec:
  eventTypes:
    - pull_request
  ownerAndRepository: mattmoor/kontext
  accessToken:
    secretKeyRef:
      name: githubsecret
      key: accessToken
  secretToken:
    secretKeyRef:
      name: githubsecret
      key: secretToken
  sink:
    apiVersion: serving.knative.dev/v1alpha1
    kind: Service
    name: github-event-display
```

```yaml
apiVersion: sources.eventing.knative.dev/v1alpha1
kind: GitHubSource
metadata:
  name: githubsourcesample
spec:
  eventTypes:
    - pull_request
  ownerAndRepository: mattmoor/kontext
  accessToken:
    secretKeyRef:
      name: githubsecret
      key: accessToken
  secretToken:
    secretKeyRef:
      name: githubsecret
      key: secretToken
  sink:
    apiVersion: eventing.knative.dev/v1alpha1
    kind: Broker
    name: vile-events
```

Where it all began: Our “Addressable” 🦆

```
apiVersion: serving.knative.dev/v1alpha1
kind: Service
metadata:
  name: foo
spec:
  ...
status:
  address:
    hostname: foo.default.svc.cluster.local
```

```
apiVersion: eventing.knative.dev/v1alpha1
kind: Broker
metadata:
  name: bar
spec:
  ...
status:
  address:
    hostname: bar.default.svc.cluster.local
```

Where it extended: Serving -> Build

Knative 0.2

```
apiVersion: serving.knative.dev/v1alpha1
kind: Revision
metadata:
  name: foo
spec:
  buildRef:
    apiVersion: build.knative.dev/v1alpha1
    kind: Build
    name: bar
status:
  ...
```

```
apiVersion: build.knative.dev/v1alpha1
kind: Build
metadata:
  name: bar
spec:
  ...
status:
  conditions:
  - type: Succeeded
    status: False
    reason: IFailed
    message: "Wesley made me do it."
```

Ducks outside: Serving is PodSpecable

Knative 0.6

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 3
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
```

```
apiVersion: serving.knative.dev/v1alpha1
kind: Service
metadata:
  name: cat-pictures
  namespace: meow
spec:
  template:
    spec:
      containers:
      - image: meow/mix:v3
  traffic:
  - name: blue
    revisionName: cat-pictures-0001
    percent: 90
  - name: green
    revisionName: cat-pictures-0002
    percent: 10
```

```
apiVersion: serving.knative.dev/v1alpha1
kind: Configuration
metadata:
  name: cat-pictures
  namespace: meow
spec:
  template:
    spec:
      containers:
      - image: meow/mix:v3
```

Ducks outside: Service + Route

Knative 0.6

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 3
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
```

```yaml
apiVersion: serving.knative.dev/v1alpha1
kind: Service
metadata:
  name: cat-pictures
  namespace: meow
spec:
  template:
    spec:
      containers:
      - image: meow/mix:v3

  traffic:
  - name: blue
    revisionName: cat-pictures-0001
    percent: 90
  - name: green
    revisionName: cat-pictures-0002
    percent: 10
```

```yaml
apiVersion: serving.knative.dev/v1alpha1
kind: Route
metadata:
  name: cat-pictures
  namespace: meow
spec:
  traffic:
  - name: blue
    revisionName: cat-pictures-0001
    percent: 90
  - name: green
    revisionName: cat-pictures-0002
    percent: 10
```

Ducks inside: Serving -> PodScalable

Knative 0.6

```
apiVersion: autoscaling.internal.knative.dev/v1alpha1
kind: PodAutoscaler
metadata:
  name: foo

spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: bar

status:
  ...
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector: ...
  template: ...

status:
  replicas: 3
  conditions: ...
```

How can you use it?

Informer Sample

```
import "k8s.io/client-go/dynamic"
import "github.com/knative/pkg/apis/duck"
...


   dynamicClient, err := dynamic.NewForConfig(cfg)
   tif := &duck.TypedInformerFactory{
      Client:       dynamicClient,
      Type:         &v1alpha1.WithPod{},
      ResyncPeriod: resyncPeriod,
      StopChannel:  stopCh,
   }
   gvr := schema.GroupVersionResource{
      Group: "serving.knative.dev",
      Version: "v1alpha1",
      Resource: "services",
   }
   informer, lister, err := tif.Get(gvr)

   informer.AddEventHandler(...)
```

```
func (c *reconciler) Reconcile(name, namespace string) error {
   untyped, err := c.lister.ByNamespace(namespace).Get(name)
   if err != nil {
      ...
   }
   thing := untyped.(*v1alpha1.WithPod)

   // Do stuff with thing!
}
```

Just waiting on Go generics… 🙄

Benefits:

- Strong typing
- Standard informer caching behavior

Patching Sample (Step 1)

```
func (ks *scaler) applyScale(ps *pav1alpha1.PodScalable, desiredScale int32) (int32, error) {
        ...

        psNew := ps.DeepCopy()
        psNew.Spec.Replicas = &desiredScale
        patch, err := duck.CreatePatch(ps, psNew)
        if err != nil {
                return desiredScale, err
        }
        patchBytes, err := patch.MarshalJSON()
        if err != nil {
                return desiredScale, err
        }


        _, err = ks.dynamicClient.Resource(*gvr).Namespace(ps.Namespace).Patch(ps.Name, types.JSONPatchType,
                patchBytes, metav1.UpdateOptions{})
```

Benefits:

- No step 2

Type Assertions

```
func TestServiceDuckTypes(t *testing.T) {
	tests := []struct {
		name string
		t    duck.Implementable
	}{{
		name: "addressable",
		t:    &duckv1alpha1.Addressable{},
	}}

	for _, test := range tests {
		t.Run(test.name, func(t *testing.T) {
			err := duck.VerifyType(&Service{}, test.t)
			if err != nil {
				t.Errorf("VerifyType(Service, %T) = %v", test.t, err)
			}
		})
	}
}
```

Duck Typing <3 “Aggregated ClusterRole”

```yaml
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
 name: knative-serving-admin
 labels:
   serving.knative.dev/release: devel
aggregationRule:
 clusterRoleSelectors:
 - matchLabels:
     serving.knative.dev/controller: "true"
rules: [] # Rules are automatically filled in
         # by the controller manager.
```

```yaml
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
 name: knative-serving-build
 labels:
   serving.knative.dev/controller: "true"
rules:
 - apiGroups: ["build.knative.dev"]
   resources: ["builds"]
   verbs: [...]
```

```yaml
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
 name: knative-serving-pipelines
 labels:
   serving.knative.dev/controller: "true"
rules:
 - apiGroups: ["tekton.dev"]
   resources: ["pipelineruns", "taskruns"]
   verbs: [...]
```

One more thing...

Where’s Istio?

```
NAMESPACE          NAME
ambassador         ambassador-7757df8b68-mp25h
ambassador         ambassador-7757df8b68-n5gld
ambassador         ambassador-7757df8b68-zxb72
boomap-system      boomap-controller-77b6d988d6-95xsp
boomap-system      webhook-67f6f69948-984fz
cachier-system     cachier-controller-668866f4dc-dr8lw
knative-serving    activator-78c8bd6c8c-mp9zp
knative-serving    autoscaler-679bfcb855-qdjhx
knative-serving    controller-68cc59764f-tglzx
```

Questions?

No ducks were harmed in the making of this presentation.

Wall of Links

- Knative:
  - slack.knative.dev
- Tools:
  - github.com/knative/pkg/tree/master/apis/duck
- Addressable:
  - github.com/knative/pkg/blob/master/apis/duck/v1beta1/addressable_types.go
- PodScalable:
  - github.com/knative/serving/blob/master/pkg/apis/autoscaling/v1alpha1/podscalable_types.go
- Demos:
  - github.com/mattmoor/cachier
  - github.com/mattmoor/boo-maps