

Kubernetes Operators made with Ansible

Michael Hrivnak

Principal Software Engineer

@michael_hrivnak

1. What is Kubernetes

```
apiVersion: v1
kind: Pod
metadata:
  name: example-app
  labels:
    app: example-app
spec:
  containers:
  - name: example
    image: companyname/example:v1.2.0
    ports:
    - containerPort: 8000
```

```
apiVersion: v1
kind: Service
metadata:
  name: example-service
spec:
  selector:
    app: example-app
  ports:
  - protocol: TCP
    port: 80
    targetPort: 8000
```

2. Ansible k8s module

K8s YAML

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: foo
  namespace: default
data:
  color: red
```

Ansible Task

```
---
- name: create foo configmap
  k8s:
    definition:
      apiVersion: v1
      kind: ConfigMap
      metadata:
        name: foo
        namespace: default
      data:
        color: "{{ color }}"
```

Ansible Template

```
---  
- name: create foo configmap  
  k8s:  
    definition: "{{ lookup('template', '/foo.yml') | from_yaml }}"
```

Ansible Role

- Packages related Ansible code for **re-use**
- Create a Role that deploys and manages your application
- Ansible Galaxy: central location to share Roles with the world

```
memcached/
├── defaults
│   └── main.yml
├── files
├── handlers
│   └── main.yml
├── meta
│   └── main.yml
├── README.md
├── tasks
│   └── main.yml
├── templates
├── tests
│   ├── inventory
│   └── test.yml
└── vars
    └── main.yml
```

Why use Ansible with Kubernetes

- Similar patterns of writing declarative/idempotent YAML
- Many ops teams are already familiar with Ansible
- Easy to learn, uses Jinja templating
- Capable of full day-2 management

3. Operators

What is an Operator?

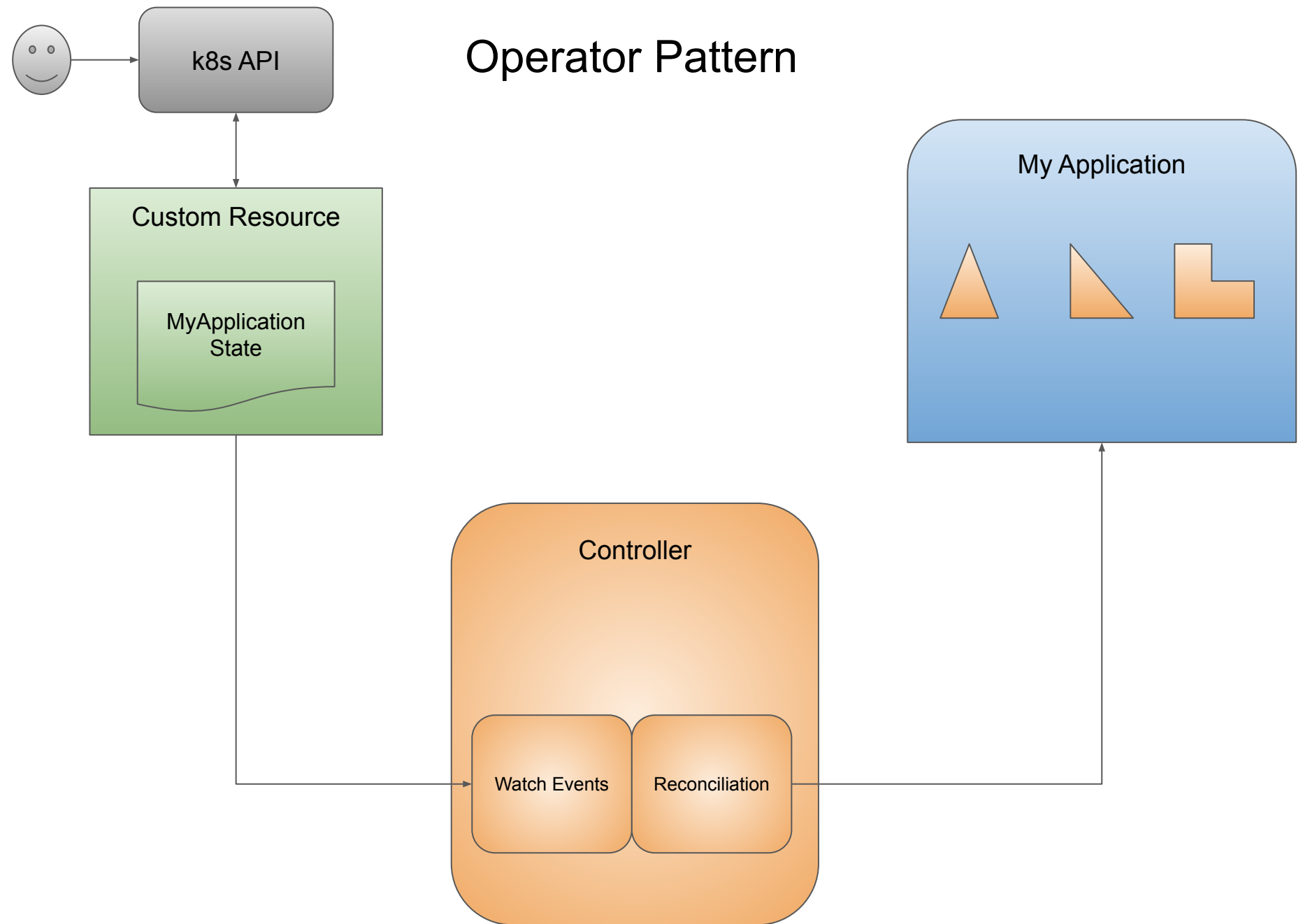
- Kubernetes Controller
- Deploys and manages an application
- Human operational knowledge in code
 - Deploy
 - Upgrade
 - Scale
 - Backup
 - Repair
 - ...

Extending the Kubernetes API

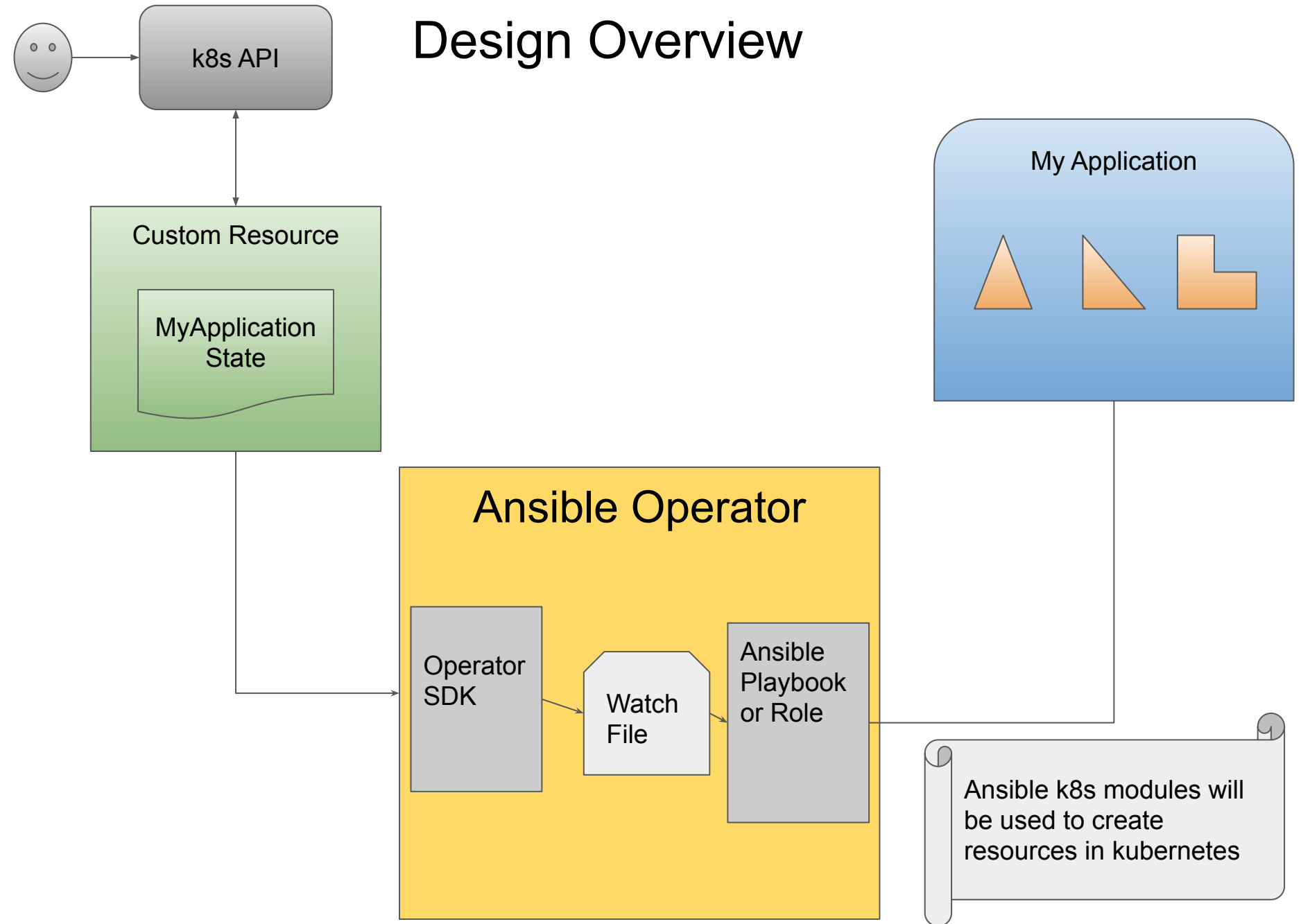
- You can define Custom Resources
- Choose what fields a user can “specify”

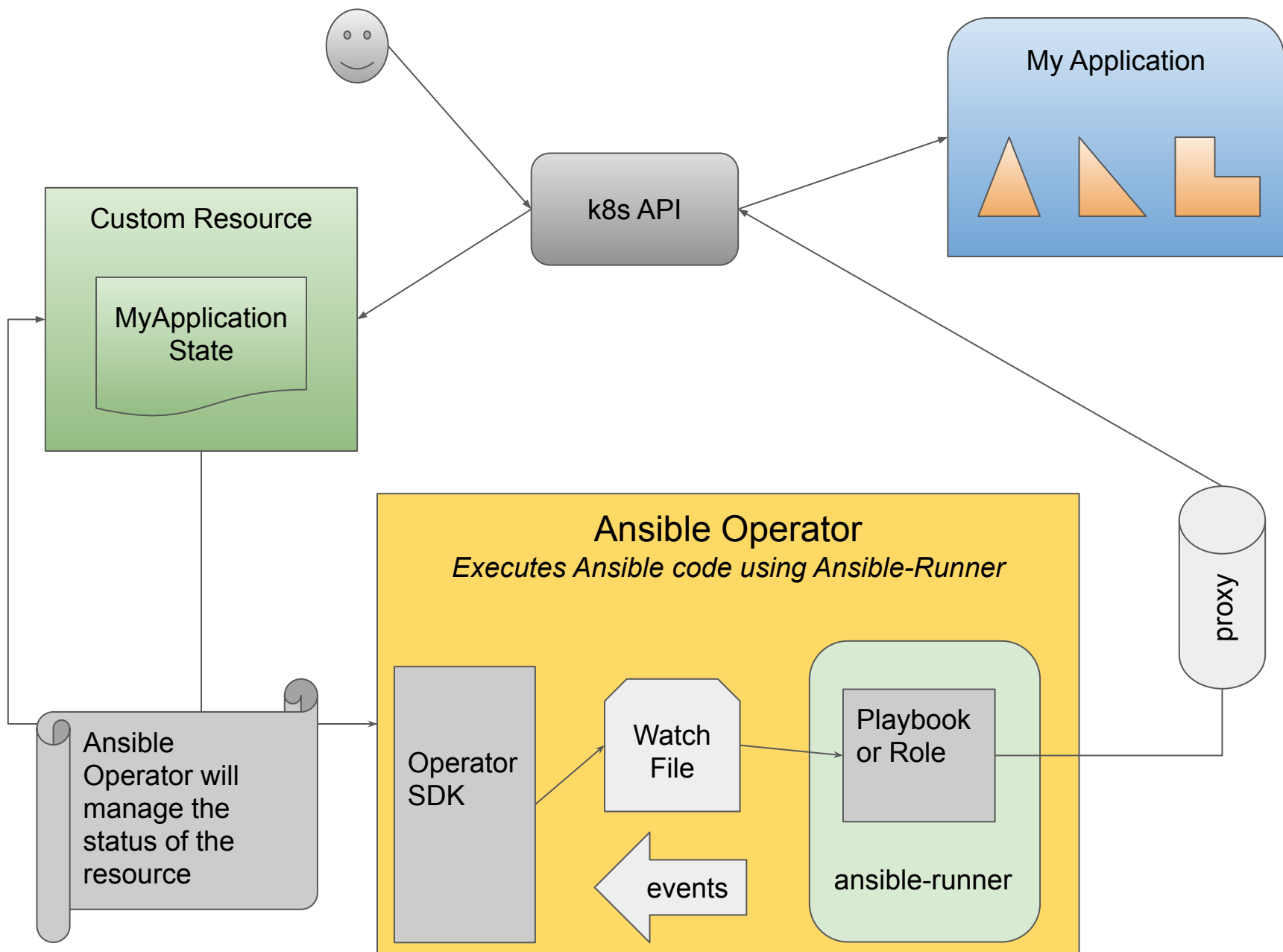
```
apiVersion: cache.example.com/v1alpha1
kind: Memcached
metadata:
  name: example-memcached
spec:
  size: 3
```

Operator Pattern



Design Overview





watches.yaml

- Maps a Group Version Kind (GVK) to a role or playbook.

```
# watches.yaml
---
- version: v1alpha1
  group: cache.example.com
  kind: Memcached
  playbook: /path/to/playbook
```



- Helps you create an operator
- Write using Go, Ansible, or Helm
- <https://github.com/operator-framework/operator-sdk/>

Spec To Parameters

Custom Resource

```
apiVersion: <Group/Version>
kind: <kind>
metadata:
  name: <name>
spec:
  <key>: <value>
  ....
status:
  <key>: <value>
  ....
```

Ansible Operator

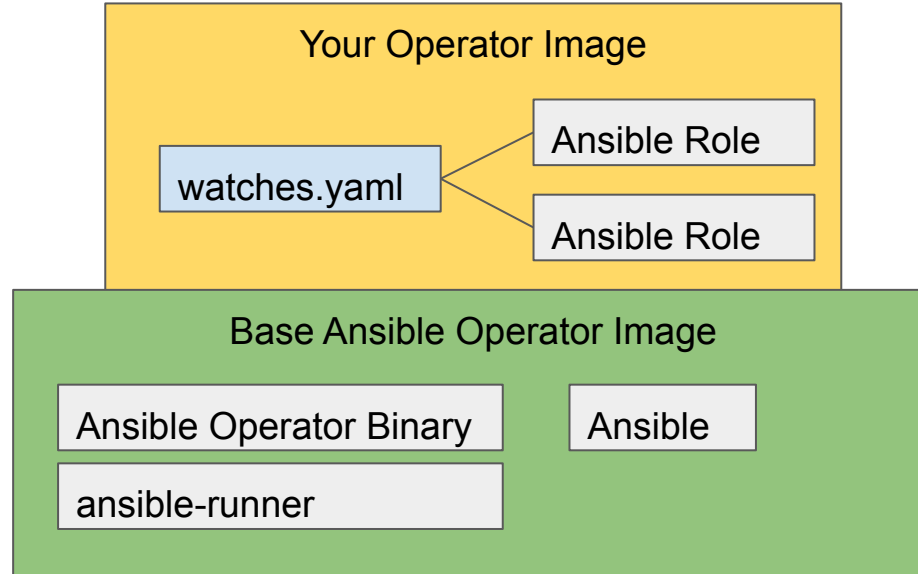
Spec values will be translated to Ansible extra vars.

Status will be a generic status defined by the operator. This will use ansible runner output to generate meaningful output for the user.

Anatomy of Operator Image

From a base Ansible Operator image:

- Add **watches.yaml**, which is a mapping of Group-Version-Kinds to a playbook or role.
- Add one or more **Ansible roles**.



Ansible Developer Experience

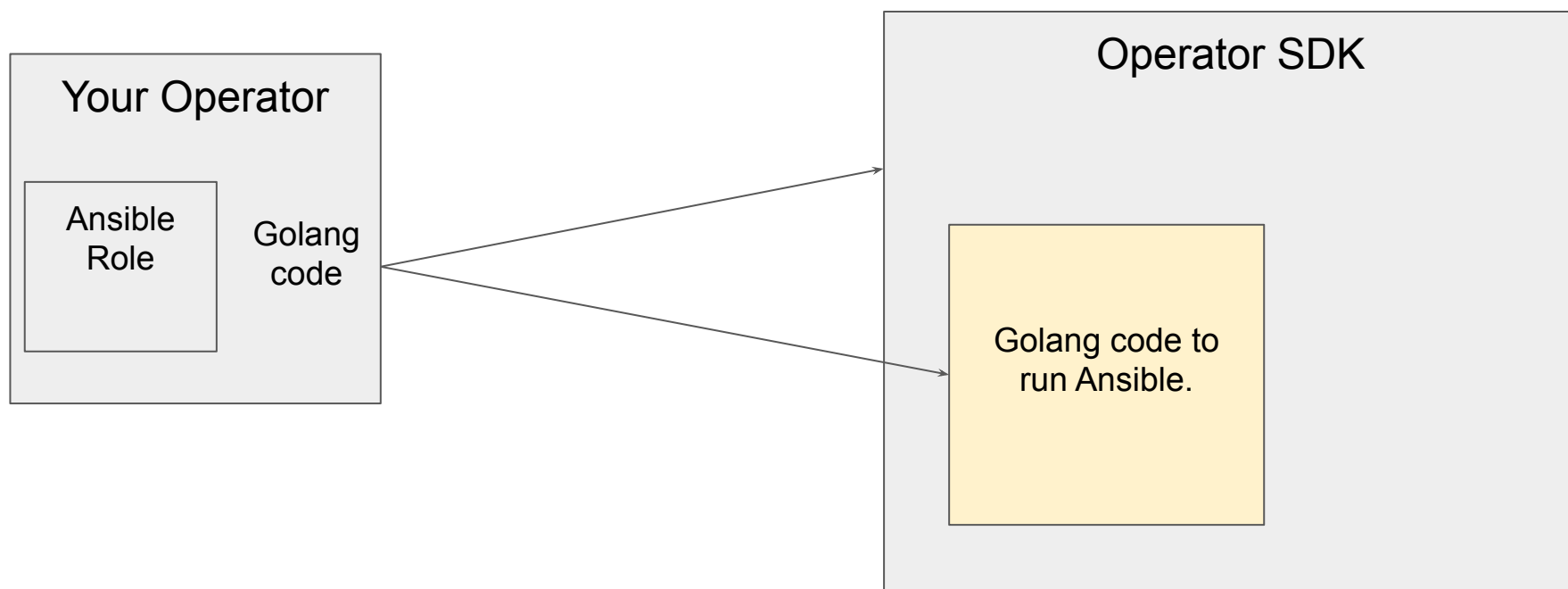
```
$ operator-sdk new memcached-operator  
--api-version=cache.example.com/v1alpha1 --kind=Memcached  
--type=ansible
```

Creates:

- Ansible Role
- Mapping File (watches.yaml)
- Custom Resource Definition
- Deploy manifest for the new Operator

Hybrid Use Case

- The Ansible Operator is a first class citizen of operator-sdk.
- You can extend/change your operator with golang code to make a hybrid.
- Allows you to change, compose, or reuse the Ansible Operator.

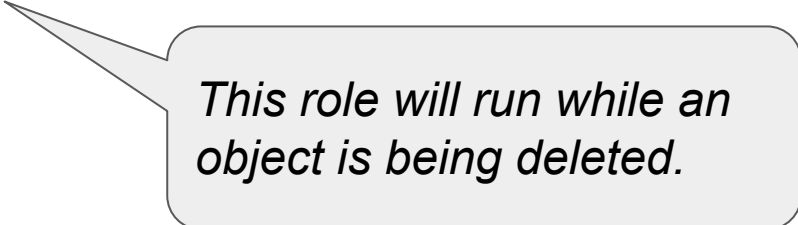


4. Advanced Patterns

Finalizers

- A way to run code before an object gets deleted.

```
# watches.yaml
---
- version: v1alpha1
  group: cache.example.com
  kind: Memcached
  role: /opt/ansible/roles/memcached
  finalizer:
    name: finalizer.memcached.cache.example.com
    role: /opt/ansible/roles/memfin
```



This role will run while an object is being deleted.

Upgrade

Can be an expression using any available variables.

- when: `version < 1.2`
block:
 - name: "run upgrade tool"
shell: `runupgrade.sh --version 1.2`
 - name: "do more upgrade stuff"
shell: ...

Backup / Restore

Create a CRD and Controller just for Backup

```
# watches.yaml
---
- version: v1beta2
  group: etcd.database.coreos.com
  kind: EtcdCluster
  playbook: /opt/ansible/playbook.yaml

- version: v1beta2
  group: etcd.database.coreos.com
  kind: EtcdBackup
  playbook: /opt/ansible/backup_playbook.yaml

- version: v1beta2
  group: etcd.database.coreos.com
  kind: EtcdRestore
  reconcilePeriod: 10h
  playbook: /opt/ansible/restore_playbook.yaml
```

Defines how and when a backup should be created.

Defines workflow logic for backup.

Questions

Michael Hrivnak
Principal Software Engineer
@michael_hrivnak