Reconciliation Pattern, Control Theory and Cluster API: The Holy Trinity

Sachin Kumar Singh
$ whoami

- Work @ Canonical (Kubernetes, MicroK8s)
- Previously worked @ VMware, working on Cluster API BYOH and upstream stuff.
- Interested in distributed systems and Cloud Native technologies.
Agenda

• First Principles: Control Theory and PID controllers (L0)
• Reconciliation Patterns in Kubernetes (L1)
• Extending Reconciliation Patterns (L2)
• Incorporating Reconciliation Patterns in Cluster API (L3)
• Demo: Cluster API MicroK8s provider
Control Theory
A simple example to understand controllers in real life..
Open-loop controllers

Timer → Dryer → Wet Clothes
A few terms

- The entity that we want to control - **System**
- The desired state - **Set Point (SP)**
- The observed state - **Process Variable (PV)**
- How “far” are we currently from our desired state? - **Error (e)**
- Who drives the system to where it needs to be? - **Controller**
closed-loop (feedback) controllers

\[ e = T_1 - T_0 \]
closed-loop (feedback) controllers

\[ e = T1 - T0 \]

System

Room

AC

T0

Process Variable (PV)

Error

set point (SP)

T1

Thermostat

Controller
However, things don’t often change fast and precisely. There’s a delay/lag when controller changes the state of the system.

So a more ideal controller would be able to account for the following:

- Undershooting or Overshooting SV.
- Compensation for large adjustments based on past experiences.
- Prediction of the future errors based on the current error.
PID controllers
PID: Propositional

The Propositional component is the linear response to the magnitude of the error.
The Integral component is the compensator. It adjusts error based on the current and past errors.
The Derivative component is the predictor. It adjusts error based on the rate of change of current errors.
If $u(t)$ is the control signal sent to the system, $y(t)$ is the measured output and $r(t)$ is the desired output, and $e(t) = r(t) - y(t)$ is the tracking error, a PID controller has the general form

$$u(t) = K_0 \ e(t) + K_1 \int_{t_1}^{t_2} e(t) \, dt + K_2 \ \frac{de(t)}{dt}$$

The desired closed loop dynamics is obtained by adjusting the three parameters $K_0$, $K_1$ and $K_2$. 
Reconciliation Patterns in Kubernetes
controller

for {
    desired := getDesiredState()
    current := getCurrentState()
    makeChanges(desired, current)
}
Deployments:
- ReplicaSet:
  - Pod0: N0
  - Pod2: N0

API server

Node 0
- Pod0
- Pod2

kubelet

Controllers
- scheduler
- deployment
- replicaset
Extending Reconciliation Patterns
// Spec defines the desired state of resource Foo

type FooSpec struct {}

// FooStatus defines the observed state of resource Foo

type FooStatus struct {}

// Foo is the Schema for the foo API

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

    Spec   FooSpec  `json:"spec,omitempty"
    Status FooStatus `json:"status,omitempty"
}

// set point

// process variable
// FooReconciler reconciles a Foo object

```go
type FooReconciler struct {
    client.Client
    Scheme *runtime.Scheme
}

func (r *FooReconciler) Reconcile(ctx context.Context, req ctrl.Request) (ctrl.Result, error) {
    // make changes to bring the current state to the desired state
    return ctrl.Result{}, nil
}
```
Custom Resource Definition (CRD)

```yaml
apiVersion: "stable.example.com/v1"
kind: Foo
metadata:
  name: foo
spec:
  ....
```
Incorporating Reconciliation Patterns in Cluster API
All these components in management cluster are custom resource managed by their respective controllers.

- **Cluster API**: Provides CAPI specific CRDs like Machines, MachineSets, Cluster etc.
- **Bootstrap Provider**: Turns a VM/server into a K8s node.
- **Control Plane Provider**: Serves the Kubernetes API and continuously reconciles desired state using reconciller loops.
- **Infrastructure Provider**: Provisions infrastructure/computational resources required by the Cluster or by Machines.
Interlude: MicroK8s

MicroK8s is a lightweight Kubernetes distribution that is designed to run on local systems.

- Automatic, autonomous and self-healing High Availability (HA).
- Comes with sensible defaults for the most widely used Kubernetes options.
- Batteries included (bring your own addons).
- CNCF certified.
Pre-demo-requisites

**Cluster API bootstrap provider MicroK8s:** Responsible for generating a cloud-init script to turn a Machine into a Kubernetes Node. This implementation uses MicroK8s for Kubernetes bootstrap.

**Cluster API control plane provider MicroK8s:** Responsible for managing the control plane of the provisioned clusters using MicroK8s.

**Cluster API infrastructure provider OpenStack:** Responsible for provisioning OpenStack VMs for clusters and nodes.
Time for a Demo
<table>
<thead>
<tr>
<th>NAME</th>
<th>CLUSTER</th>
<th>NODENAME</th>
<th>PROVIDERID</th>
<th>PHASE</th>
<th>AGE</th>
<th>VERSION</th>
</tr>
</thead>
<tbody>
<tr>
<td>test-ci-cluster-control-plane-klp9k</td>
<td>test-ci-cluster</td>
<td>test-ci-cluster-control-plane-rngm</td>
<td>openstack:///43217b72-31b8-475f-89b7-49b22944b760</td>
<td>Running</td>
<td>5m42s</td>
<td>v1.24.0</td>
</tr>
<tr>
<td>test-ci-cluster-control-md-0-5ccf575597-7fks2</td>
<td>test-ci-cluster</td>
<td>test-ci-cluster-control-plane-rngm</td>
<td>openstack:///43217b72-31b8-475f-89b7-49b22944b760</td>
<td>Provisioning</td>
<td>6m42s</td>
<td>v1.24.0</td>
</tr>
<tr>
<td>test-ci-cluster-control-md-0-5ccf575597-tbhuz</td>
<td>test-ci-cluster</td>
<td>test-ci-cluster-control-plane-rngm</td>
<td>openstack:///43217b72-31b8-475f-89b7-49b22944b760</td>
<td>Provisioning</td>
<td>6m42s</td>
<td>v1.24.0</td>
</tr>
<tr>
<td>test-ci-cluster-control-md-0-5ccf575597-wqrfv</td>
<td>test-ci-cluster</td>
<td>test-ci-cluster-control-plane-mvqtp</td>
<td>openstack:///43217b72-31b8-475f-89b7-49b22944b760</td>
<td>Provisioning</td>
<td>6m42s</td>
<td>v1.24.0</td>
</tr>
<tr>
<td>test-ci-cluster-control-plane-t8wgb</td>
<td>test-ci-cluster</td>
<td>test-ci-cluster-control-plane-rngm</td>
<td>openstack:///43217b72-31b8-475f-89b7-49b22944b760</td>
<td>Provisioning</td>
<td>5m42s</td>
<td>v1.24.0</td>
</tr>
<tr>
<td>NAME</td>
<td>CLUSTER</td>
<td>NODENAME</td>
<td>PROVIDERID</td>
<td>PHASE</td>
<td>AGE</td>
<td>VERSION</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>------------------</td>
<td>---------------------------------</td>
<td>----------------------------------</td>
<td>----------</td>
<td>-------</td>
<td>---------</td>
</tr>
<tr>
<td>test-ci-cluster-control-plane-kpm9x</td>
<td>test-ci-cluster</td>
<td>test-ci-cluster-control-plane-rngm</td>
<td>openstack://42217872-3170-475f-80b7-49b2a9b44b760</td>
<td>Running</td>
<td>7m9s</td>
<td>v1.24.0</td>
</tr>
<tr>
<td>test-ci-cluster-md-0.5ccf575597-wqrfv</td>
<td>test-ci-cluster</td>
<td>test-ci-cluster-control-plane-rngm</td>
<td>openstack://492f6fd8-3d12-49a8-955b-65ff811e77ae</td>
<td>Provisioned</td>
<td>8m9s</td>
<td>v1.24.0</td>
</tr>
<tr>
<td>test-ci-cluster-md-0.5ccf575597-t8hrz</td>
<td>test-ci-cluster</td>
<td>test-ci-cluster-control-plane-rngm</td>
<td>openstack://39d430d5-c532-4a28-b7e9-173208111053</td>
<td>Provisioned</td>
<td>8m9s</td>
<td>v1.24.0</td>
</tr>
<tr>
<td>test-ci-cluster-control-plane-nvtep</td>
<td>test-ci-cluster</td>
<td>test-ci-cluster-control-plane-rngm</td>
<td>openstack://366f1f1c-245c-4b01-9bab-931299a0a492</td>
<td>Provisioned</td>
<td>7m9s</td>
<td>v1.24.0</td>
</tr>
<tr>
<td>test-ci-cluster-control-plane-nvtep</td>
<td>test-ci-cluster</td>
<td>test-ci-cluster-control-plane-rngm</td>
<td>openstack://37b01910-429b-4f73-8563-ad6ff9ee2e69</td>
<td>Provisioned</td>
<td>8m9s</td>
<td>v1.24.0</td>
</tr>
<tr>
<td>test-ci-cluster-control-plane-t8wgb</td>
<td>test-ci-cluster</td>
<td>test-ci-cluster-control-plane-rngm</td>
<td>openstack://2b2d6de2-6d7-4831-aad4-4ca3dbb89f2e</td>
<td>Provisioned</td>
<td>7m9s</td>
<td>v1.24.0</td>
</tr>
<tr>
<td>NAME</td>
<td>CLUSTER</td>
<td>NODENAME</td>
<td>PROVIDERID</td>
<td>PHASE</td>
<td>AGE</td>
<td>VERSION</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>---------------------</td>
<td>----------------------</td>
<td>-------------------------------------------------</td>
<td>--------</td>
<td>-------</td>
<td>---------</td>
</tr>
<tr>
<td>test-0-0-0.5ccf575597-tk9frz</td>
<td>test-cl-cluster</td>
<td>test-cl-cluster-0-Aqdp7</td>
<td>openstack:///fba483b-629b-4757-b063-a68f53ce3ed9</td>
<td>Running</td>
<td>23m</td>
<td>v1.24.0</td>
</tr>
<tr>
<td>test-0-0.5ccf575597-tk9frz</td>
<td>test-cl-cluster</td>
<td>test-cl-cluster-0-qrpw6</td>
<td>openstack:///39d43d5-d53e-4a28-b7e9-173268116053</td>
<td>Running</td>
<td>23m</td>
<td>v1.24.0</td>
</tr>
<tr>
<td>test-0-0.5ccf575597-wgrfv</td>
<td>test-cl-cluster</td>
<td>test-cl-cluster-0-nyppq</td>
<td>openstack:///6921f6f8-3d12-49a8-9b5b-65ff8111e9f7aa</td>
<td>Running</td>
<td>23m</td>
<td>v1.24.0</td>
</tr>
<tr>
<td>test-0-0.5ccf575597-tbwgb</td>
<td>test-cl-cluster</td>
<td>test-cl-cluster-0-gw9wp</td>
<td>openstack:///5b2e6e2-4dd7-4311-aadc-4ca3dbd57f2e</td>
<td>Running</td>
<td>23m</td>
<td>v1.24.0</td>
</tr>
<tr>
<td>test-0-0.5ccf575597-xwp9x</td>
<td>test-cl-cluster</td>
<td>test-cl-cluster-0-xgppb</td>
<td>openstack:///f70778be-5f41-4e99-8b63-9df0dc88260</td>
<td>Running</td>
<td>67s</td>
<td>v1.24.0</td>
</tr>
<tr>
<td>NAME</td>
<td>CLUSTER</td>
<td>NODENAME</td>
<td>PROVIDERID</td>
<td>PHASE</td>
<td>AGE</td>
<td>VERSION</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>-----------------------</td>
<td>-----------------------------</td>
<td>-----------------------------</td>
<td>------------</td>
<td>-------</td>
<td>----------</td>
</tr>
<tr>
<td>test-cl-cluster-nd-0-5c9f575597-tdhrz</td>
<td>test-cl-cluster</td>
<td>test-cl-cluster-nd-0-qrpwk</td>
<td>openstack://29d489d5-c332-4a2b-b7e9-172265911983</td>
<td>Running</td>
<td>26m</td>
<td>v1.24.0</td>
</tr>
<tr>
<td>test-cl-cluster-nd-0-5c9f575597-wqtfv</td>
<td>test-cl-cluster</td>
<td>test-cl-cluster-nd-0-nhppq</td>
<td>openstack://692f66fd-8d12-49ab-9b5b-65ff8116f7ae</td>
<td>Running</td>
<td>26m</td>
<td>v1.24.0</td>
</tr>
<tr>
<td>test-cl-cluster-control-plane-t8wpb</td>
<td>test-cl-cluster</td>
<td>test-cl-cluster-control-plane-gw9wp</td>
<td>openstack://2ba7dc2-e6d7-4811-aadc-4ca3dbb99f7c</td>
<td>Running</td>
<td>26m</td>
<td>v1.24.0</td>
</tr>
<tr>
<td>test-cl-cluster-control-plane-xq6w96</td>
<td>test-cl-cluster</td>
<td>test-cl-cluster-control-plane-rngm</td>
<td>openstack://43217672-3176-43af-97b9-40b229445a6d</td>
<td>Running</td>
<td>26m</td>
<td>v1.24.0</td>
</tr>
<tr>
<td>test-cl-cluster-control-plane-lq8skp</td>
<td>test-cl-cluster</td>
<td>test-cl-cluster-control-plane-xqpm</td>
<td>openstack://70778bce-1541-4e99-8b63-69ff00c88260</td>
<td>Running</td>
<td>26m</td>
<td>v1.24.0</td>
</tr>
<tr>
<td>test-cl-cluster-nd-0-7bcbbc5c5-tx9x</td>
<td>test-cl-cluster-nd-0-qdlyq</td>
<td>test-cl-cluster-nd-0-4dv24</td>
<td>openstack://52ae03f1-0bf2-4134-979b-0877287ac162</td>
<td>Running</td>
<td>26m</td>
<td>v1.24.0</td>
</tr>
<tr>
<td>test-cl-cluster-nd-0-5c9f575597-9rk52</td>
<td>test-cl-cluster-nd-0-8qds6</td>
<td>test-cl-cluster-nd-0-8qds6</td>
<td>openstack://7081938-029b-4787-b803-6a8053ee2ed9</td>
<td>Deleting</td>
<td>26m</td>
<td>v1.24.0</td>
</tr>
<tr>
<td>NAME</td>
<td>CLUSTER</td>
<td>NODENAME</td>
<td>PROVIDERID</td>
<td>PHASE</td>
<td>AGE</td>
<td>VERSION</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>---------------</td>
<td>-------------------</td>
<td>-----------------------------</td>
<td>------------</td>
<td>------</td>
<td>---------</td>
</tr>
<tr>
<td>test-0.cluster-nd-0.5ccf675597-tskrz</td>
<td>test-0.cluster</td>
<td>test-0.cluster-nd-0.qqqw</td>
<td>openstack://239d238d5-c322-4a3b-9be4-37f269111003</td>
<td>Running</td>
<td>34m</td>
<td>v1.24.0</td>
</tr>
<tr>
<td>test-0.cluster-nd-0.5ccf675597-tskrz</td>
<td>test-0.cluster</td>
<td>test-0.cluster-nd-0.nhggg</td>
<td>openstack://690f6f6b-30f2-4f3b-9f5b-65f0f35e77fa</td>
<td>Running</td>
<td>34m</td>
<td>v1.24.0</td>
</tr>
<tr>
<td>test-0.cluster-nd-0.7bc5c35c-tbqfw</td>
<td>test-0.cluster</td>
<td>test-0.cluster-nd-0.qppm</td>
<td>openstack://20f0778be-1e41-4e99-86f3-96f000c88200</td>
<td>Running</td>
<td>10m</td>
<td>v1.24.0</td>
</tr>
<tr>
<td>test-0.cluster-nd-0.7bc5c35c-tbqfw</td>
<td>test-0.cluster</td>
<td>test-0.cluster-nd-0.jkdbm</td>
<td>openstack://21e6044-61f2-4e3b-a850-2beb7853896</td>
<td>Running</td>
<td>9m47s</td>
<td>v1.24.0</td>
</tr>
<tr>
<td>test-0.cluster-nd-0.7bc5c35c-tbqfw</td>
<td>test-0.cluster</td>
<td>test-0.cluster-nd-0.vqz4</td>
<td>openstack://62e0df1-0bf2-4334-9780-b08727ac1b2</td>
<td>Running</td>
<td>9m47s</td>
<td>v1.24.0</td>
</tr>
<tr>
<td>test-0.cluster-nd-0.7bc5c35c-tbqfw</td>
<td>test-0.cluster</td>
<td>test-0.cluster-nd-0.tsxq</td>
<td>openstack://6e2e163-07c9-45da-bf70-c3977308214</td>
<td>Running</td>
<td>5m33s</td>
<td>v1.24.0</td>
</tr>
<tr>
<td>test-0.cluster-nd-0.7bc5c35c-tbqfw</td>
<td>test-0.cluster</td>
<td>test-0.cluster-nd-0.spldn</td>
<td>openstack://779c4c2-1d43-4090-99e3-76ec46c384ca</td>
<td>Running</td>
<td>5m33s</td>
<td>v1.24.0</td>
</tr>
<tr>
<td>test-0.cluster-nd-0.7bc5c35c-tbqfw</td>
<td>test-0.cluster</td>
<td>test-0.cluster-nd-0.rnqmm</td>
<td>openstack://43217872-3179-4f5f-89b7-49b2e94487d0</td>
<td>Deleting</td>
<td>33m</td>
<td>v1.24.0</td>
</tr>
</tbody>
</table>
References and Resources

- Cluster API book
- Control Theory in Container Fleet Management
- Control Theory is Dope
- Close Loops & Opening Minds: How to Take Control of Systems, Big & Small
- Controller Architecture Kubernetes docs
- Kubebuilder Book
- MicroK8s Bootstrap provider docs
- MicroK8s official docs
- Control Theory, Controllers and Kubernetes: The Holy Trinity
Thank you!

Twitter: @sachin_singh092
GitHub: @sachinkumarsingh092
K8s slack: @sachinkumarsingh092 (#microk8s)