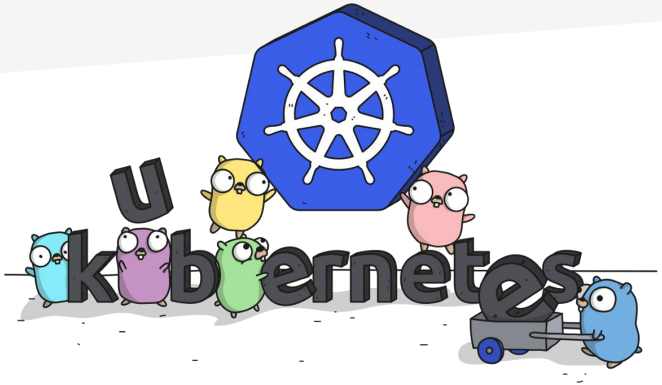


Reconciliation Pattern, Control Theory and Cluster API: The Holy Trinity

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.

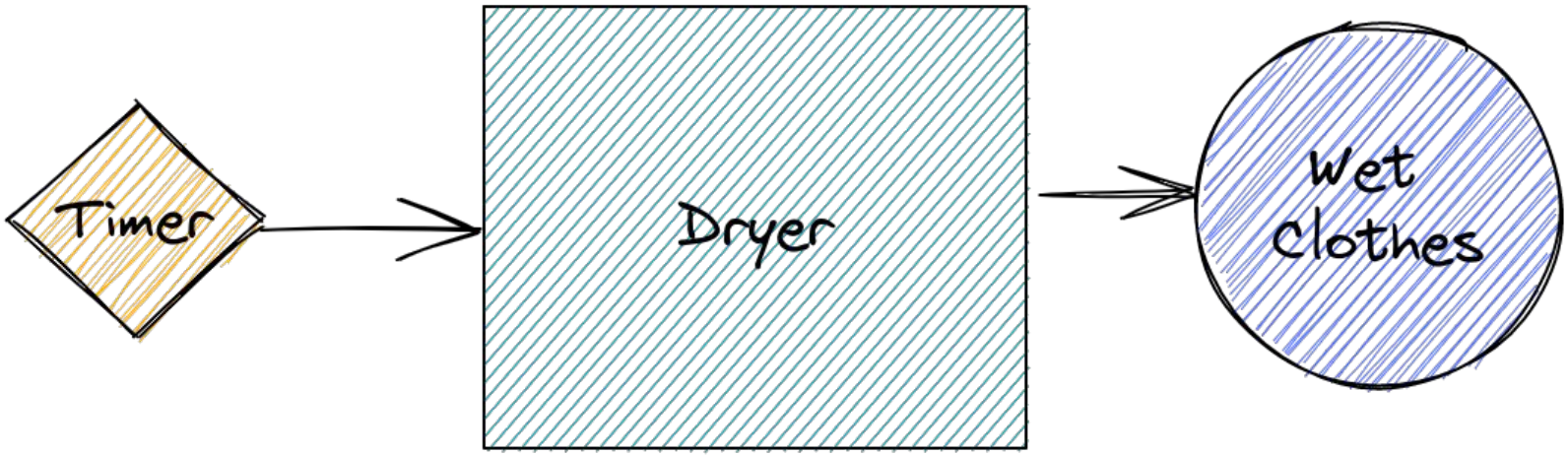
Agenda

- 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..

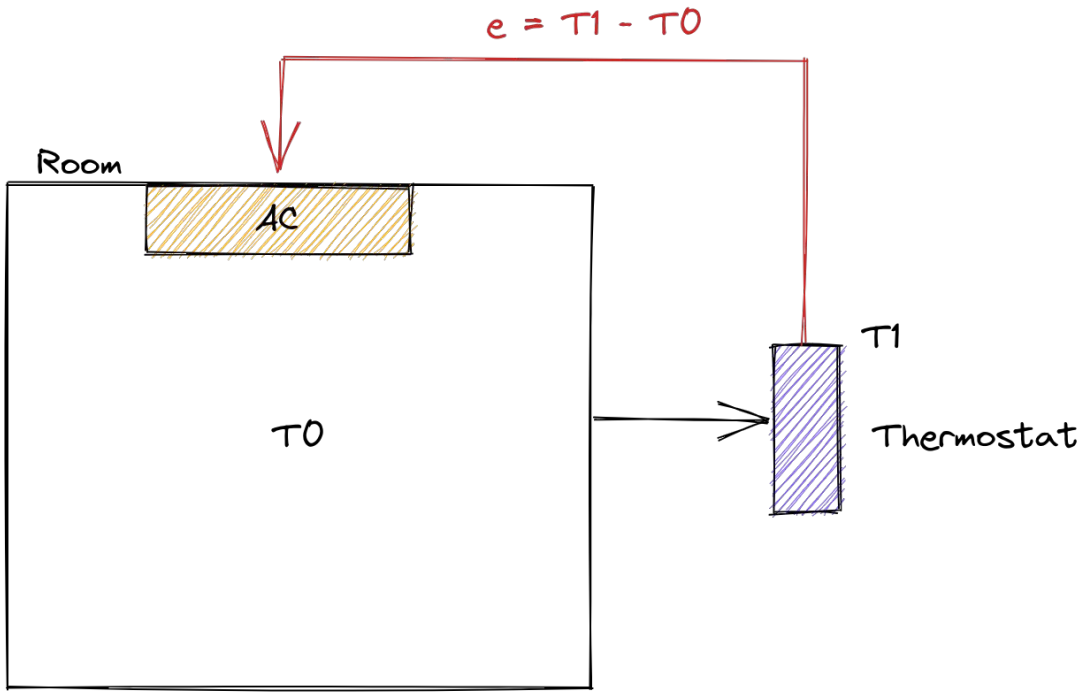
Open-loop controllers



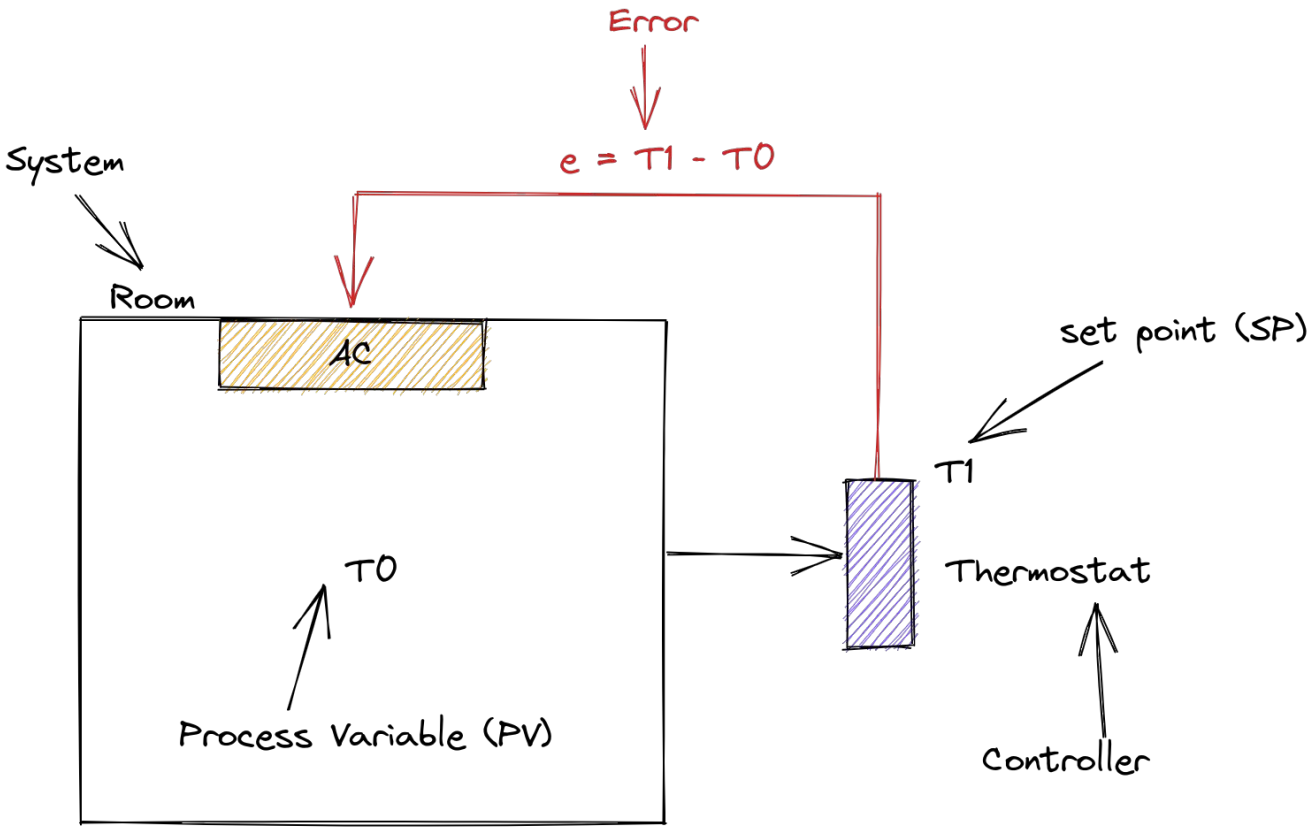
A few terms

- 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



closed-loop (feedback) controllers



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

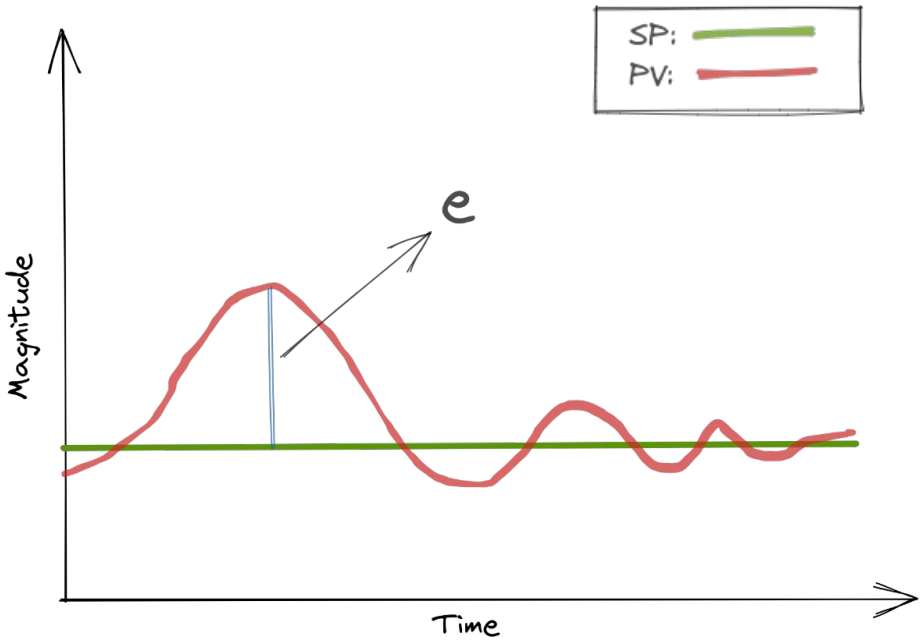
So a more ideal controller would be able to account for the following:

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

PID controllers

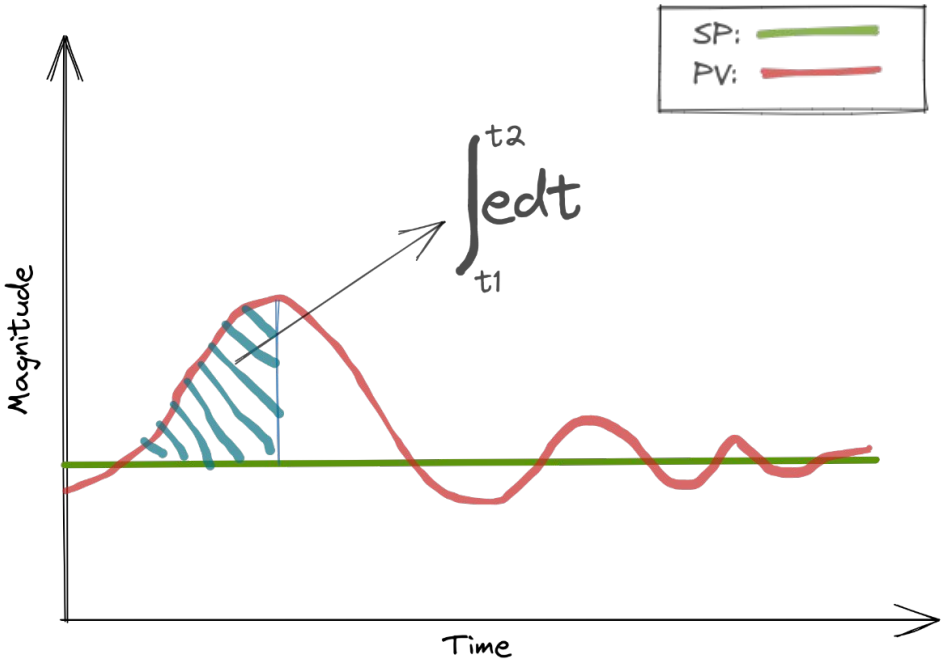
PID: Propositional

The Propositional component is the linear response to the magnitude of the error



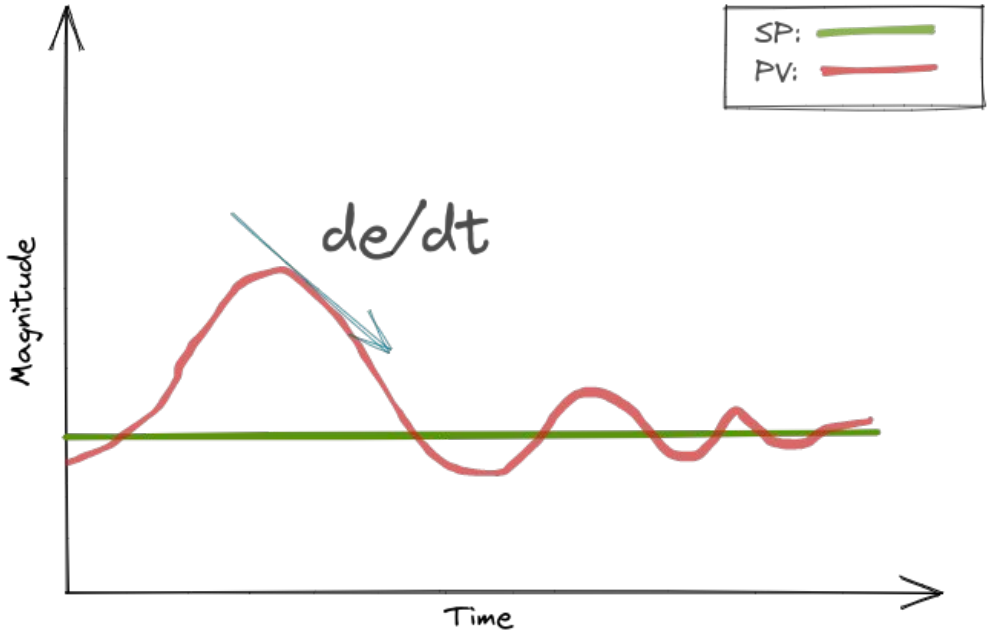
PID: Integral

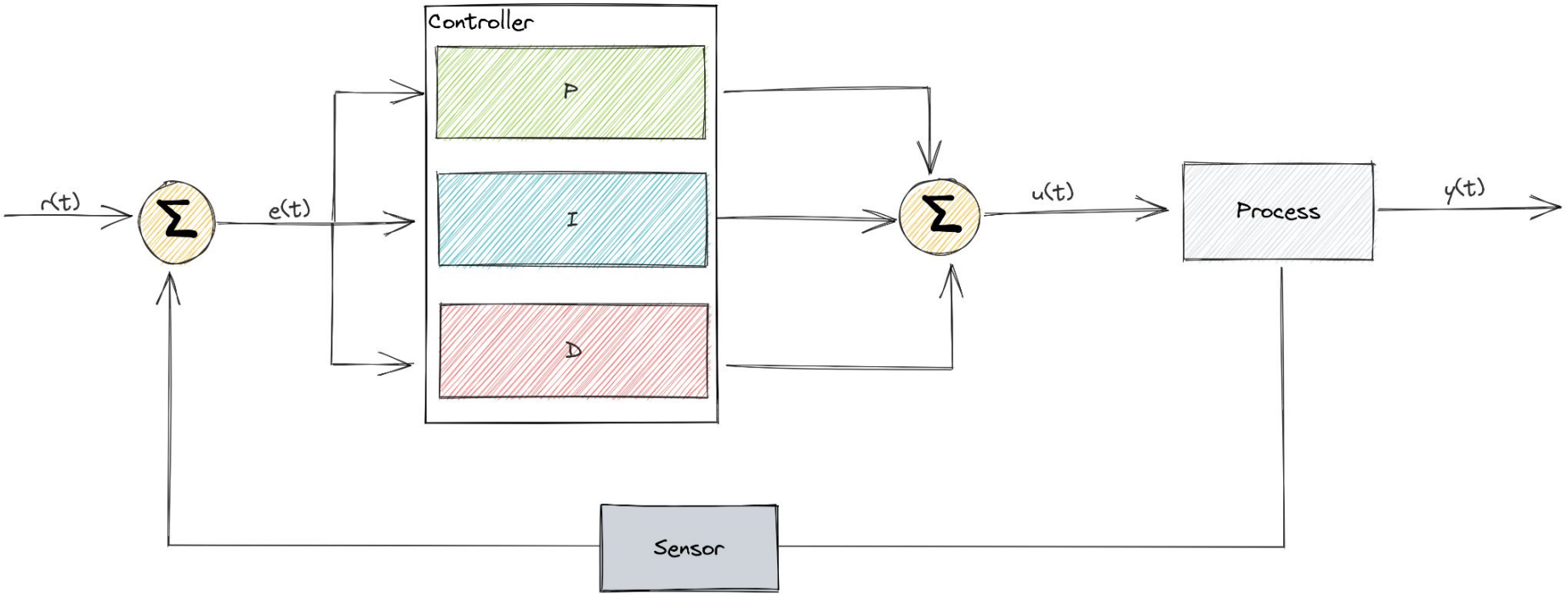
The Integral component is the compensator. It adjust error based on the current and past errors.



PID: Derivative

The Derivative component is the predictor. It adjusts error based on the rate of change of current errors.



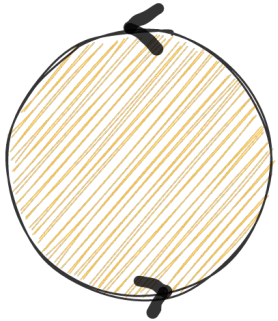

Controller
P
I
D
r(t)
e(t)
u(t)
Process
y(t)
Sensor

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

$$u(t) = K0\ e(t) + K1 \int_{t1}^{t2} e(t)dt + K2\ de(t)/dt$$

The desired closed loop dynamics is obtained by adjusting the three parameters K0, K1 and K2.

Reconciliation Patterns in Kubernetes



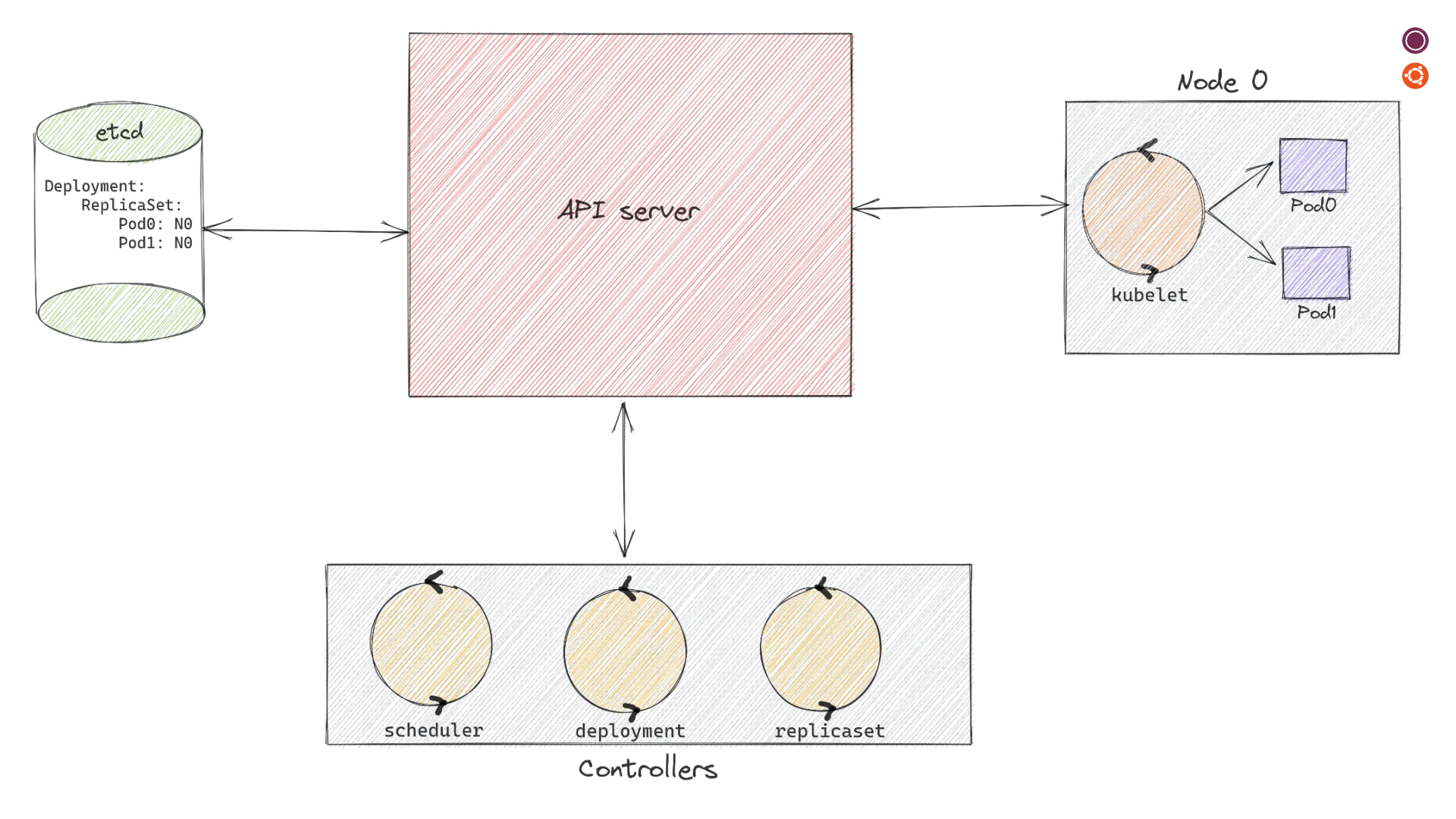
controller

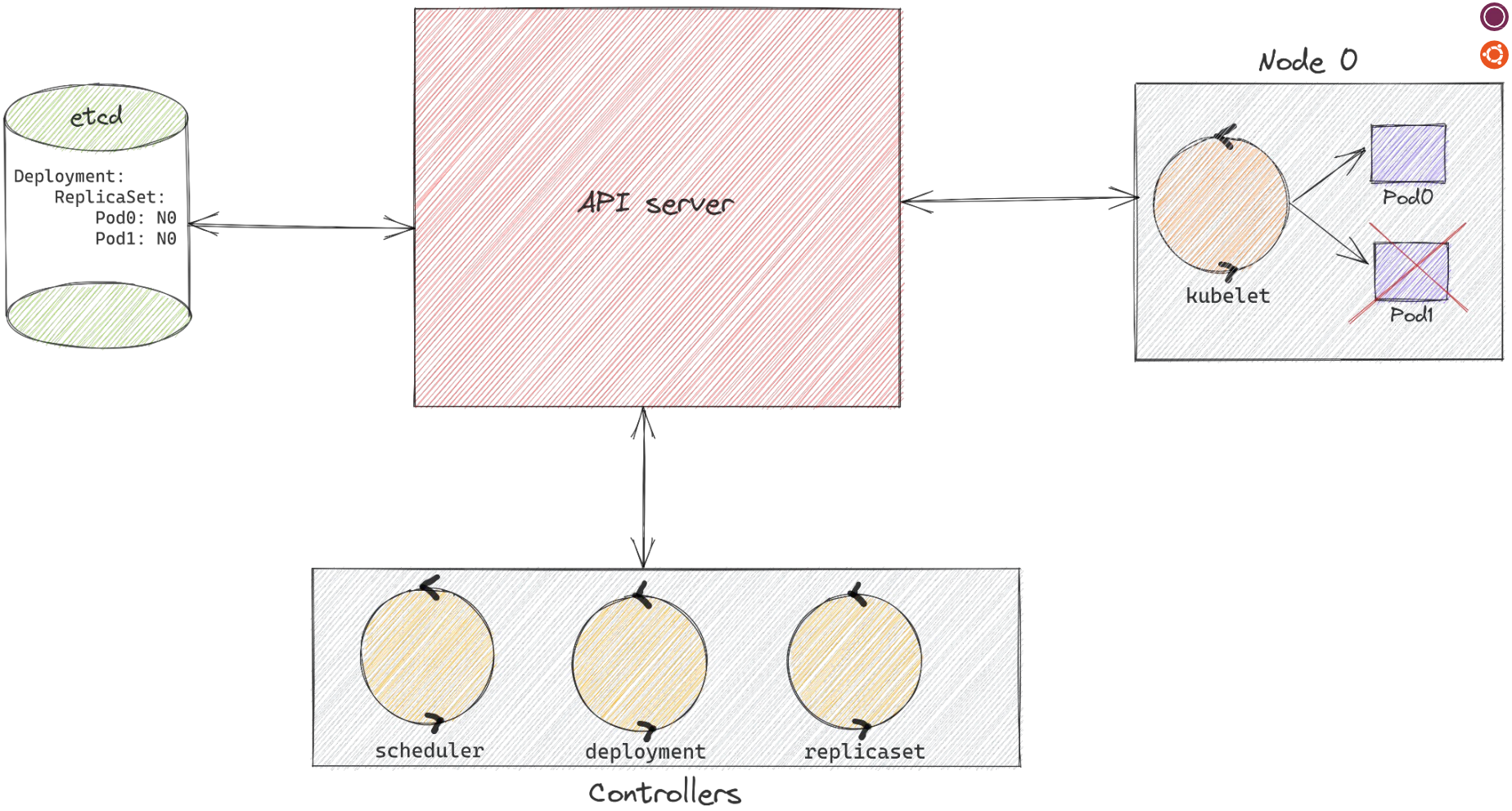
```
for {
  desired := getDesiredState()
  current := getCurrentState()
  makeChanges(desired, current)
}
```

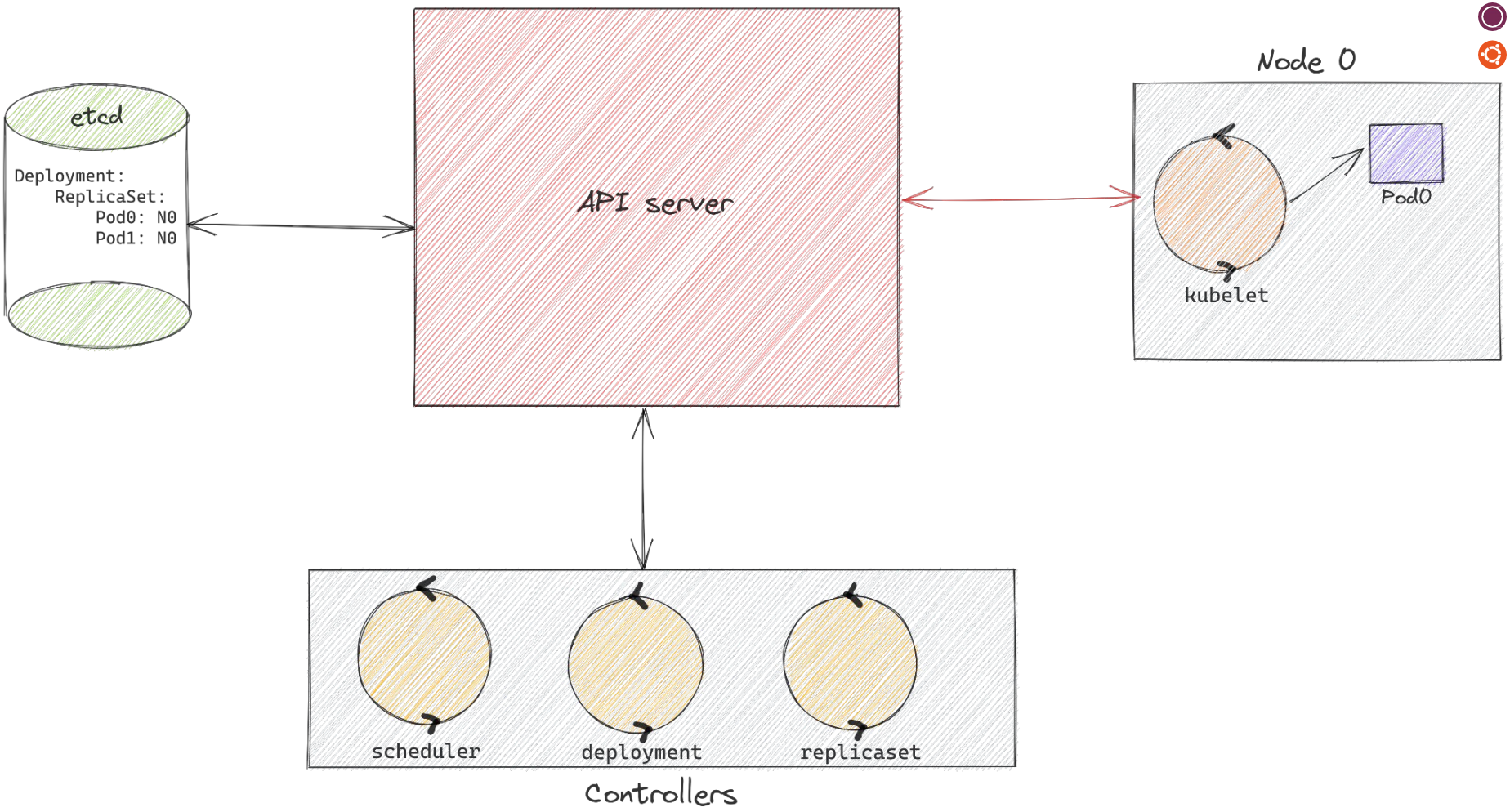
set point

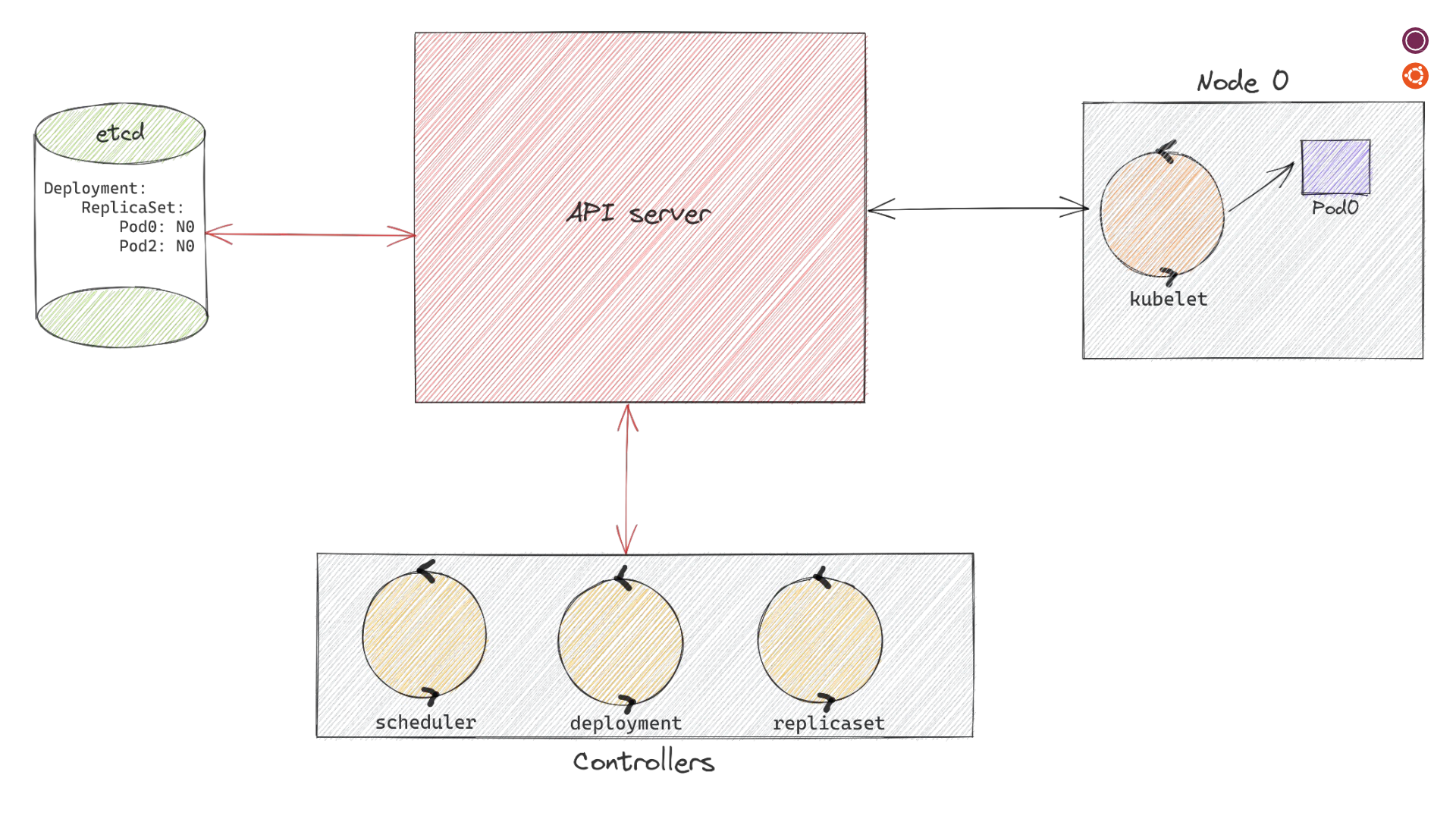
process variable

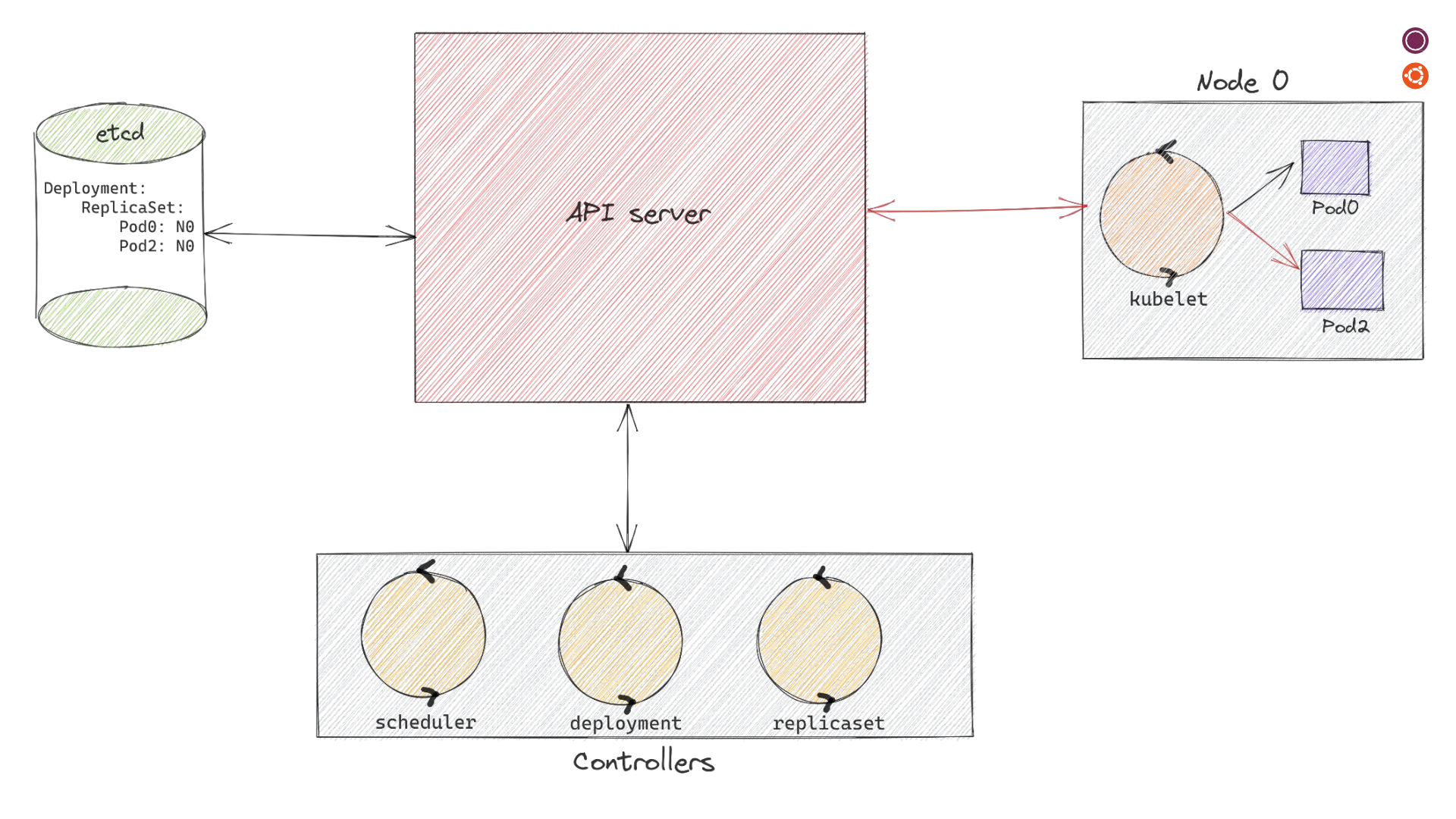
https://github.com/kubernetes/community/blob/master/contributors/devel/sig-api-machinery/controllers.md


etcd
Deployment:
ReplicaSet:
Pod0: N0
Pod1: N0
API server
Node 0
kubelet
Pod0
Pod1
scheduler
deployment
replicaset
Controllers


etcd
Deployment:
ReplicaSet:
Pod0: N0
Pod1: N0
API server
Node 0
kubelet
Pod0
Pod1
scheduler
deployment
replicaset
Controllers


etcd
Deployment:
ReplicaSet:
Pod0: N0
Pod1: N0
API server
Node 0
kubelet
Pod0
scheduler
deployment
replicaset
Controllers


etcd
Deployment:
ReplicaSet:
Pod0: N0
Pod2: N0
API server
Node 0
kubelet
Pod0
scheduler
deployment
replicaset
Controllers


etcd
Deployment:
ReplicaSet:
Pod0: N0
Pod2: N0
API server
Node 0
kubelet
Pod0
Pod2
scheduler
deployment
replicaset
Controllers

Extending Reconciliation Patterns

```
// Spec defines the desired state of resource Foo
type FooSpec struct {}
```

set point

```
// FooStatus defines the observed state of resource Foo
type FooStatus struct {}
```

process variable

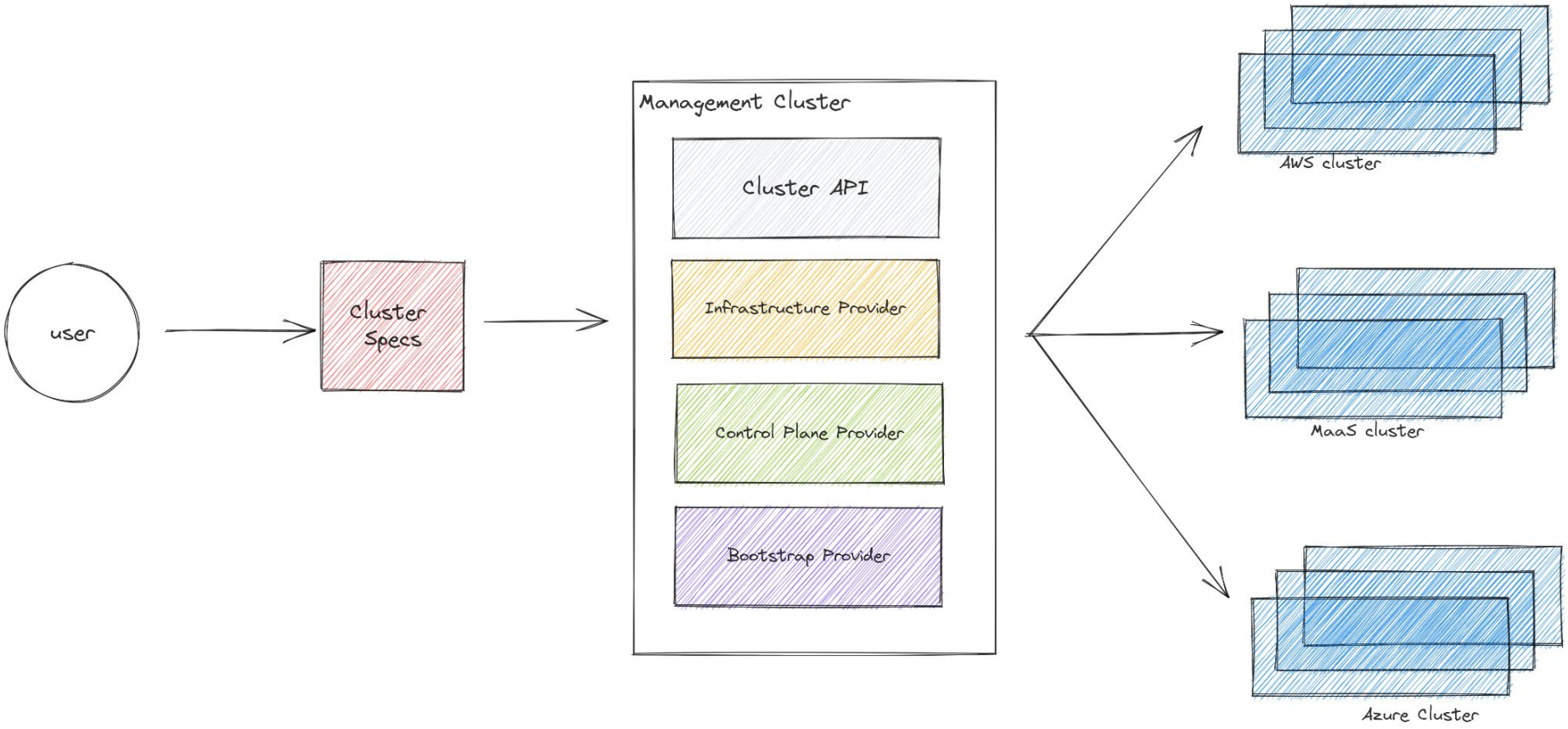
```
// Foo is the Schema for the foo API
type Foo struct {
    metav1.TypeMeta   `json:",inline"`
    metav1.ObjectMeta `json:"metadata,omitempty"`

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

```
// FooReconciler reconciles a Foo object
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)

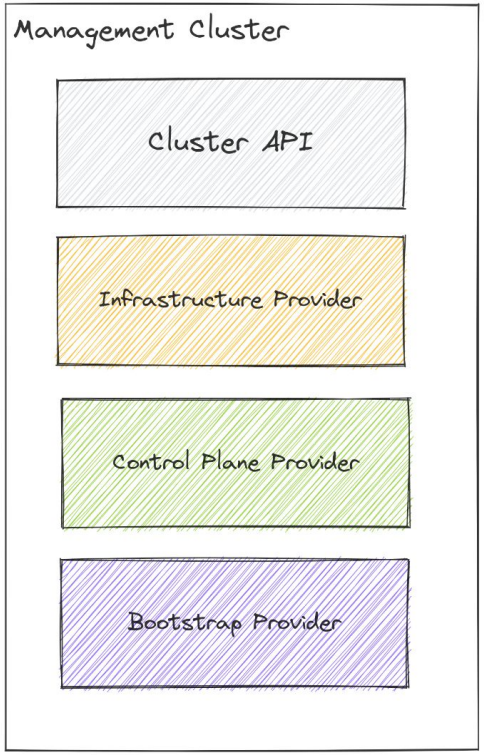
```
apiVersion: "stable.example.com/v1"
kind: Foo
metadata:
  name: foo
spec:
  ....
```

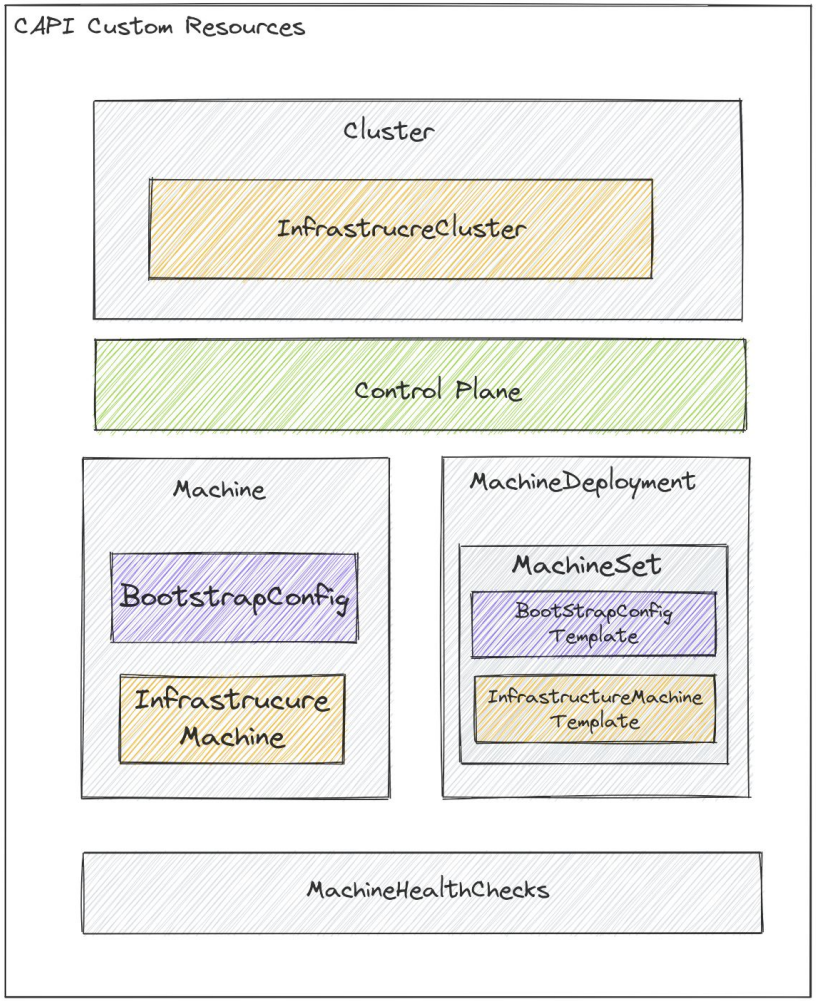
Incorporating Reconciliation Patterns in Cluster API


user
Cluster Specs
Management Cluster
Cluster API
Infrastructure Provider
Control Plane Provider
Bootstrap Provider
AWS cluster
MaaS cluster
Azure Cluster

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.


Management Cluster
Cluster API
Infrastructure Provider
Control Plane Provider
Bootstrap Provider


CAPI Custom Resources
Cluster
InfrastrucreCluster
Control Plane
Machine
BootstrapConfig
Infrastrucure Machine
MachineDeployment
MachineSet
BootStrapConfig Template
InfrastructureMachine Template
MachineHealthChecks

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.

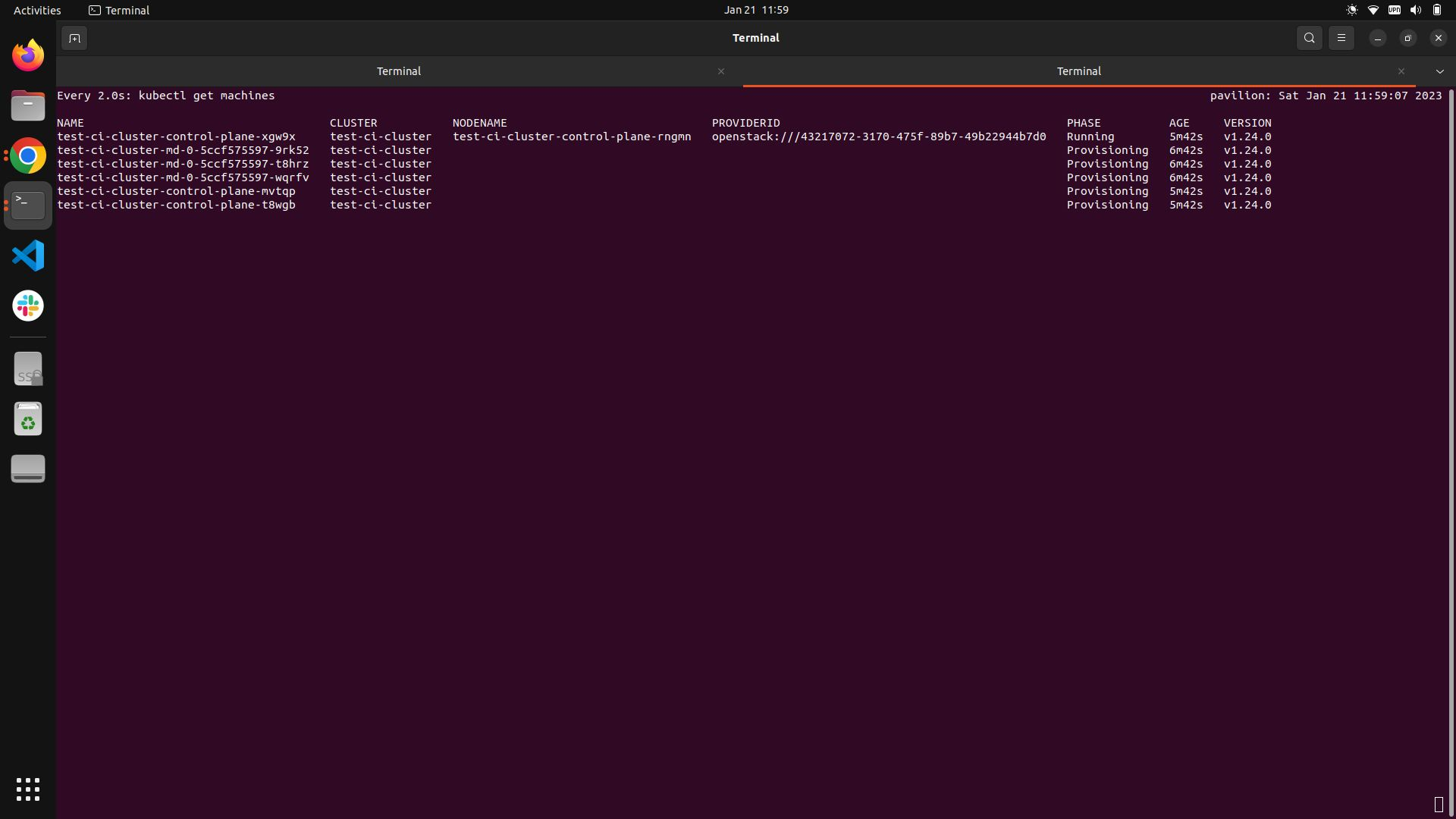
Pre-demo-requisites

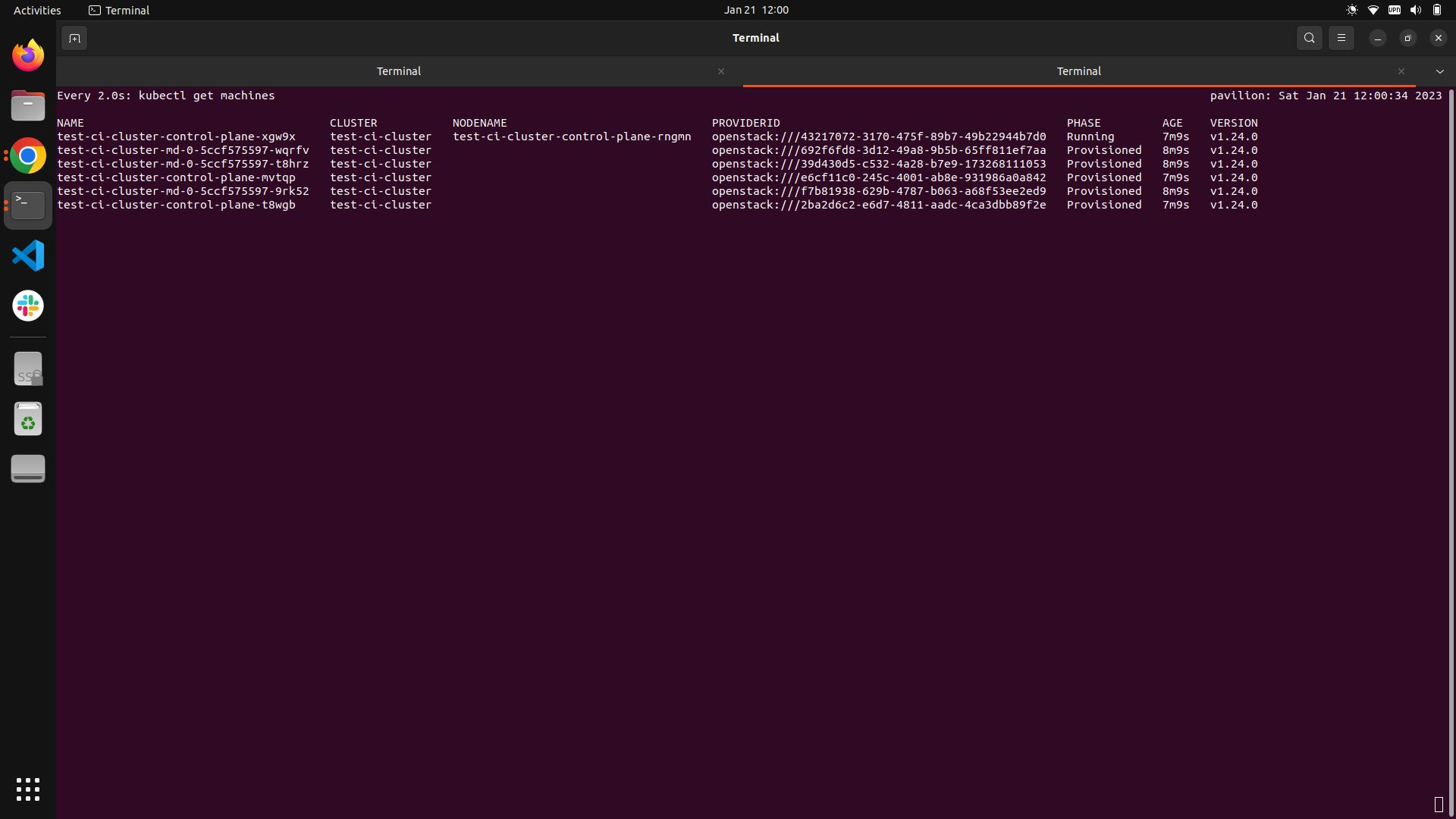
Cluster API bootstrap provider MicroK8s: Responsible for generating a cloud-init script to turn a Machine into a Kubernetes Node. This implementation uses MicroK8s for Kubernetes bootstrap.

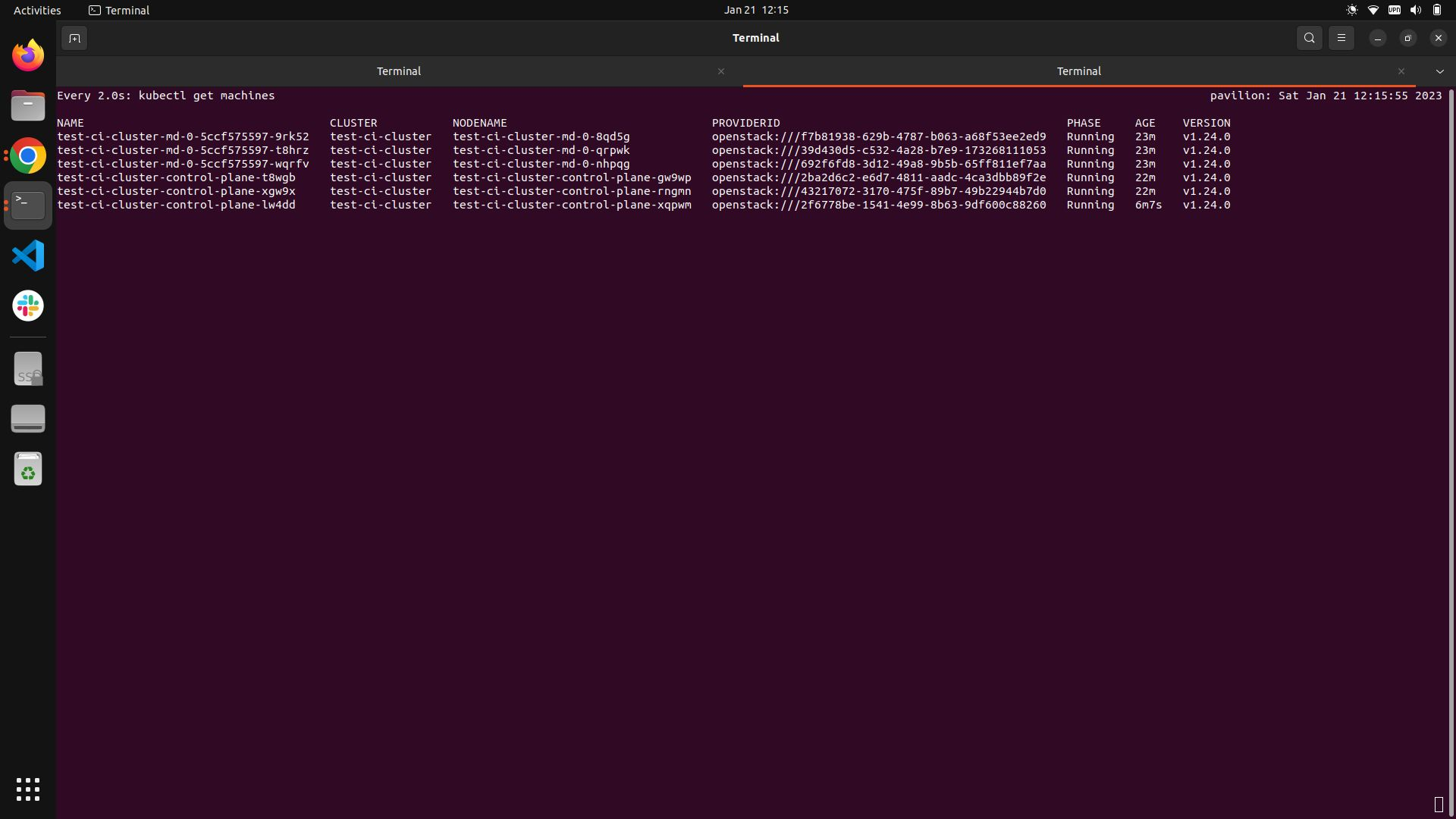
Cluster API control plane provider MicroK8s: Responsible for managing the control plane of the provisioned clusters using MicroK8s.

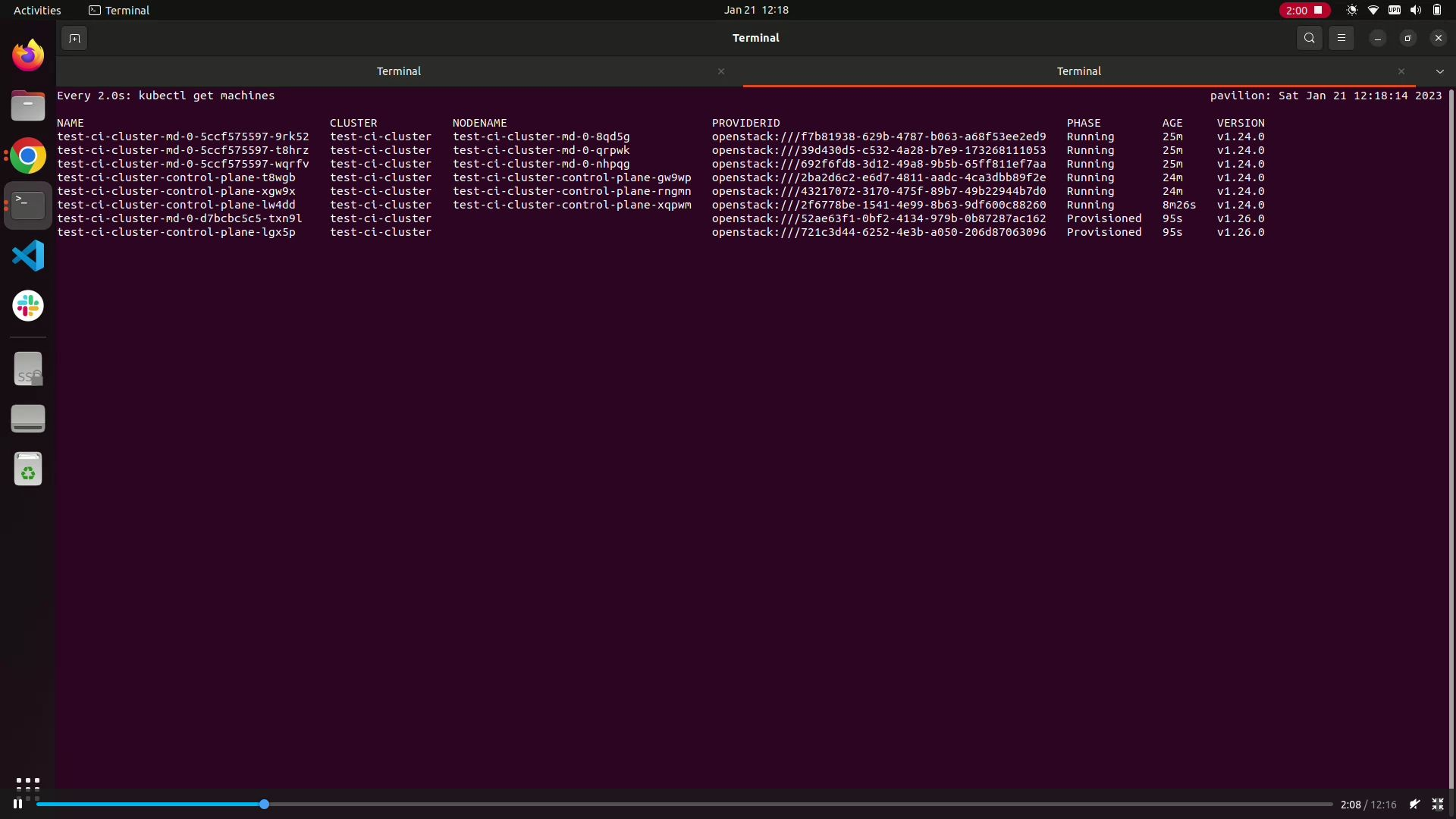
Cluster API infrastructure provider OpenStack: Responsible for provisioning OpenStack VMs for clusters and nodes.

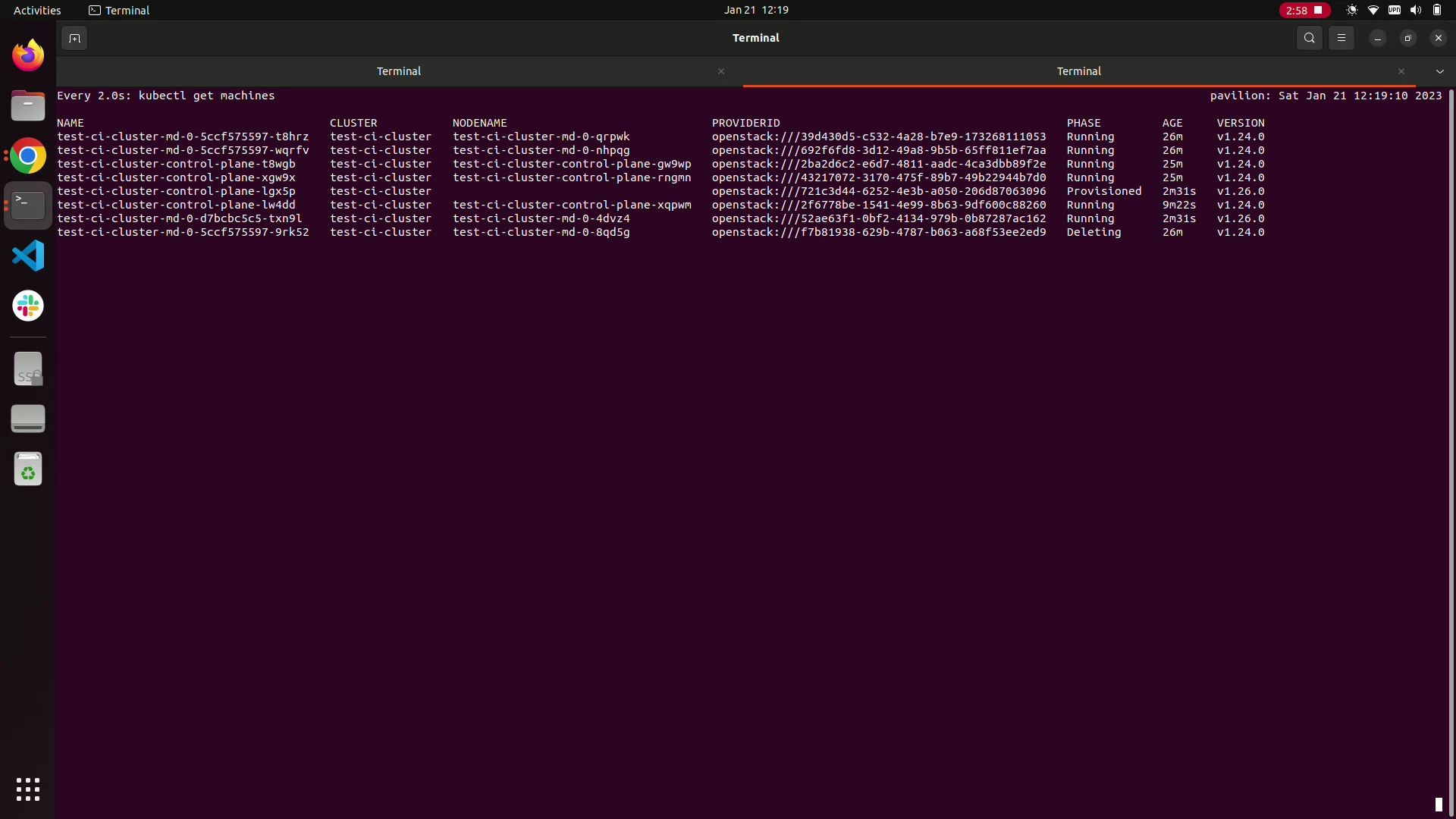
Time for a Demo

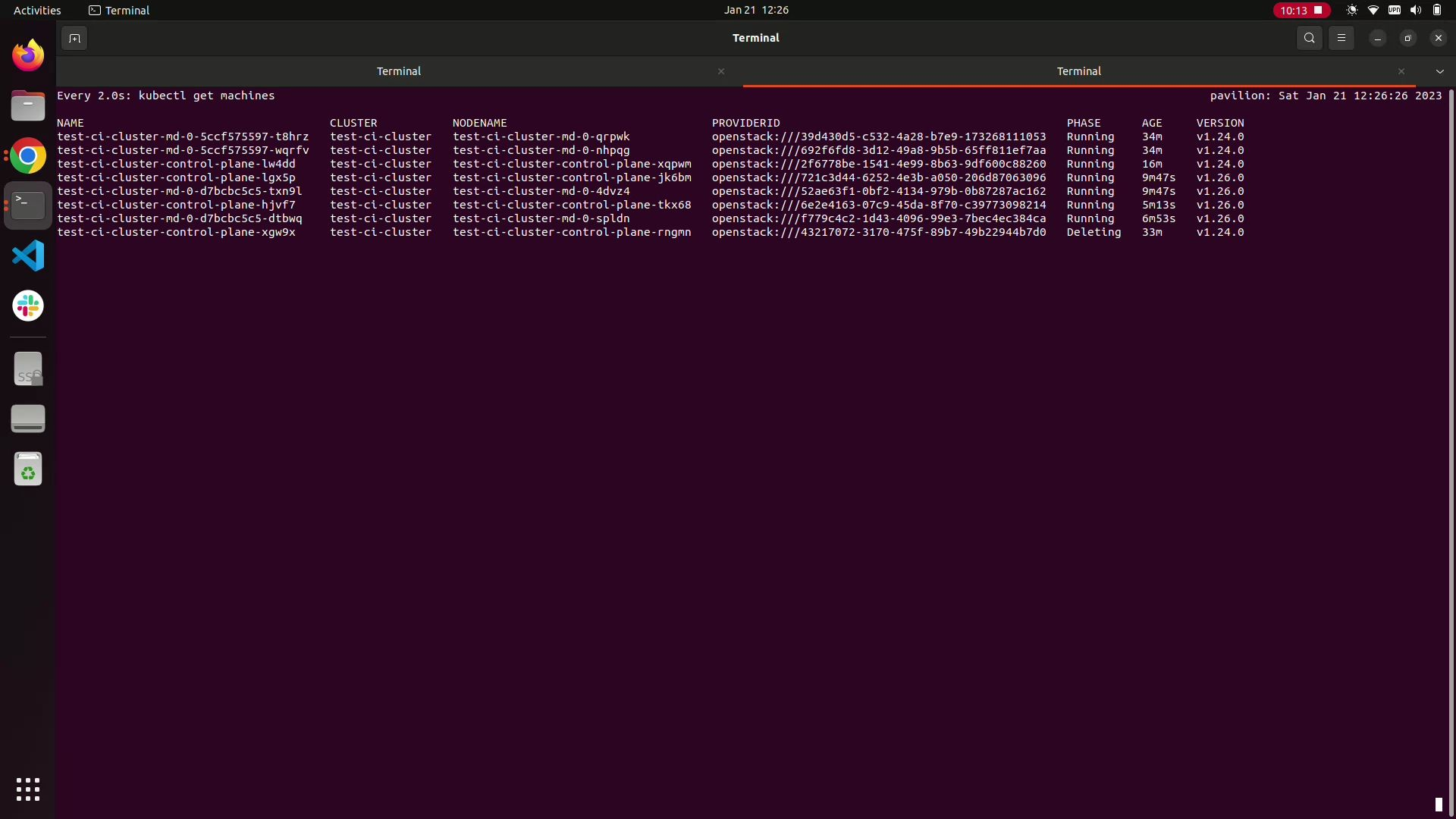

Activities
Terminal
Jan 21 11:59
Terminal
Terminal
Terminal
Every 2.0s: kubectl get machines
pavilion: Sat Jan 21 11:59:07 2023
NAME CLUSTER NODENAME PROVIDERID PHASE AGE VERSION
test-ci-cluster-control-plane-xgw9x test-ci-cluster test-ci-cluster-control-plane-rngmn openstack:///43217072-3170-475f-89b7-49b22944b7d0 Running 5m42s v1.24.0
test-ci-cluster-md-0-5ccf575597-9rk52 test-ci-cluster Provisioning 6m42s v1.24.0
test-ci-cluster-md-0-5ccf575597-t8hrz test-ci-cluster Provisioning 6m42s v1.24.0
test-ci-cluster-md-0-5ccf575597-wqrfv test-ci-cluster Provisioning 6m42s v1.24.0
test-ci-cluster-control-plane-mvtqp test-ci-cluster Provisioning 5m42s v1.24.0
test-ci-cluster-control-plane-t8wgb test-ci-cluster Provisioning 5m42s v1.24.0


Activities
Terminal
Jan 21 12:00
Terminal
Terminal
Terminal
Every 2.0s: kubectl get machines
pavilion: Sat Jan 21 12:00:34 2023
NAME CLUSTER NODENAME PROVIDERID PHASE AGE VERSION
test-ci-cluster-control-plane-xgw9x test-ci-cluster test-ci-cluster-control-plane-rngmn openstack:///43217072-3170-475f-89b7-49b22944b7d0 Running 7m9s v1.24.0
test-ci-cluster-md-0-5ccf575597-wqrfv test-ci-cluster openstack:///692f6fd8-3d12-49a8-9b5b-65ff811ef7aa Provisioned 8m9s v1.24.0
test-ci-cluster-md-0-5ccf575597-t8hrz test-ci-cluster openstack:///39d430d5-c532-4a28-b7e9-173268111053 Provisioned 8m9s v1.24.0
test-ci-cluster-control-plane-mvtqp test-ci-cluster openstack:///e6cf11c0-245c-4001-ab8e-931986a0a842 Provisioned 7m9s v1.24.0
test-ci-cluster-md-0-5ccf575597-9rk52 test-ci-cluster openstack:///f7b81938-629b-4787-b063-a68f53ee2ed9 Provisioned 8m9s v1.24.0
test-ci-cluster-control-plane-t8wgb test-ci-cluster openstack:///2ba2d6c2-e6d7-4811-aadc-4ca3dbb89f2e Provisioned 7m9s v1.24.0


Activities
Terminal
Jan 21 12:15
Terminal
Terminal
Terminal
Every 2.0s: kubectl get machines
pavilion: Sat Jan 21 12:15:55 2023
NAME CLUSTER NODENAME PROVIDERID PHASE AGE VERSION
test-ci-cluster-md-0-5ccf575597-9rk52 test-ci-cluster test-ci-cluster-md-0-8qd5g openstack:///f7b81938-629b-4787-b063-a68f53ee2ed9 Running 23m v1.24.0
test-ci-cluster-md-0-5ccf575597-t8hrz test-ci-cluster test-ci-cluster-md-0-qrpwk openstack:///39d430d5-c532-4a28-b7e9-173268111053 Running 23m v1.24.0
test-ci-cluster-md-0-5ccf575597-wqrfv test-ci-cluster test-ci-cluster-md-0-nhpqg openstack:///692f6fd8-3d12-49a8-9b5b-65ff811ef7aa Running 23m v1.24.0
test-ci-cluster-control-plane-t8wgb test-ci-cluster test-ci-cluster-control-plane-gw9wp openstack:///2ba2d6c2-e6d7-4811-aadc-4ca3dbb89f2e Running 22m v1.24.0
test-ci-cluster-control-plane-xgw9x test-ci-cluster test-ci-cluster-control-plane-rngmn openstack:///43217072-3170-475f-89b7-49b22944b7d0 Running 22m v1.24.0
test-ci-cluster-control-plane-lw4dd test-ci-cluster test-ci-cluster-control-plane-xqpwm openstack:///2f6778be-1541-4e99-8b63-9df600c88260 Running 6m7s v1.24.0


Activities
Terminal
Jan 21 12:18
2:00
Terminal
Terminal
Terminal
Every 2.0s: kubectl get machines
pavilion: Sat Jan 21 12:18:14 2023
NAME CLUSTER NODENAME PROVIDERID PHASE AGE VERSION
test-ci-cluster-md-0-5ccf575597-9rk52 test-ci-cluster test-ci-cluster-md-0-8qd5g openstack:///f7b81938-629b-4787-b063-a68f53ee2ed9 Running 25m v1.24.0
test-ci-cluster-md-0-5ccf575597-t8hrz test-ci-cluster test-ci-cluster-md-0-qrpwk openstack:///39d430d5-c532-4a28-b7e9-173268111053 Running 25m v1.24.0
test-ci-cluster-md-0-5ccf575597-wqrfv test-ci-cluster test-ci-cluster-md-0-nhpqg openstack:///692f6fd8-3d12-49a8-9b5b-65ff811ef7aa Running 25m v1.24.0
test-ci-cluster-control-plane-t8wgb test-ci-cluster test-ci-cluster-control-plane-gw9wp openstack:///2ba2d6c2-e6d7-4811-aadc-4ca3dbb89f2e Running 24m v1.24.0
test-ci-cluster-control-plane-xgw9x test-ci-cluster test-ci-cluster-control-plane-rngmn openstack:///43217072-3170-475f-89b7-49b22944b7d0 Running 24m v1.24.0
test-ci-cluster-control-plane-lw4dd test-ci-cluster test-ci-cluster-control-plane-xqpwm openstack:///2f6778be-1541-4e99-8b63-9df600c88260 Running 8m26s v1.24.0
test-ci-cluster-md-0-d7bcbc5c5-txn9l test-ci-cluster openstack:///52ae63f1-0bf2-4134-979b-0b87287ac162 Provisioned 95s v1.26.0
test-ci-cluster-control-plane-lgx5p test-ci-cluster openstack:///721c3d44-6252-4e3b-a050-206d87063096 Provisioned 95s v1.26.0
2:08 / 12:16


Activities
Terminal
Jan 21 12:19
2:58
Terminal
Terminal
Terminal
Every 2.0s: kubectl get machines
pavilion: Sat Jan 21 12:19:10 2023
NAME CLUSTER NODENAME PROVIDERID PHASE AGE VERSION
test-ci-cluster-md-0-5ccf575597-t8hrz test-ci-cluster test-ci-cluster-md-0-qrpwk openstack:///39d430d5-c532-4a28-b7e9-173268111053 Running 26m v1.24.0
test-ci-cluster-md-0-5ccf575597-wqrfv test-ci-cluster test-ci-cluster-md-0-nhpqg openstack:///692f6fd8-3d12-49a8-9b5b-65ff811ef7aa Running 26m v1.24.0
test-ci-cluster-control-plane-t8wgb test-ci-cluster test-ci-cluster-control-plane-gw9wp openstack:///2ba2d6c2-e6d7-4811-aadc-4ca3dbb89f2e Running 25m v1.24.0
test-ci-cluster-control-plane-xgw9x test-ci-cluster test-ci-cluster-control-plane-rngmn openstack:///43217072-3170-475f-89b7-49b22944b7d0 Running 25m v1.24.0
test-ci-cluster-control-plane-lgx5p test-ci-cluster openstack:///721c3d44-6252-4e3b-a050-206d87063096 Provisioned 2m31s v1.26.0
test-ci-cluster-control-plane-lw4dd test-ci-cluster test-ci-cluster-control-plane-xqpwm openstack:///2f6778be-1541-4e99-8b63-9df600c88260 Running 9m22s v1.24.0
test-ci-cluster-md-0-d7bcbc5c5-txn9l test-ci-cluster test-ci-cluster-md-0-4dvz4 openstack:///52ae63f1-0bf2-4134-979b-0b87287ac162 Running 2m31s v1.26.0
test-ci-cluster-md-0-5ccf575597-9rk52 test-ci-cluster test-ci-cluster-md-0-8qd5g openstack:///f7b81938-629b-4787-b063-a68f53ee2ed9 Deleting 26m v1.24.0


Activities
Terminal
Jan 21 12:26
10:13
Terminal
Terminal
Terminal
Every 2.0s: kubectl get machines
pavilion: Sat Jan 21 12:26:26 2023
NAME CLUSTER NODENAME PROVIDERID PHASE AGE VERSION
test-ci-cluster-md-0-5ccf575597-t8hrz test-ci-cluster test-ci-cluster-md-0-qrpwk openstack:///39d430d5-c532-4a28-b7e9-173268111053 Running 34m v1.24.0
test-ci-cluster-md-0-5ccf575597-wqrfv test-ci-cluster test-ci-cluster-md-0-nhpqg openstack:///692f6fd8-3d12-49a8-9b5b-65ff811ef7aa Running 34m v1.24.0
test-ci-cluster-control-plane-lw4dd test-ci-cluster test-ci-cluster-control-plane-xqpwm openstack:///2f6778be-1541-4e99-8b63-9df600c88260 Running 16m v1.24.0
test-ci-cluster-control-plane-lgx5p test-ci-cluster test-ci-cluster-control-plane-jk6bm openstack:///721c3d44-6252-4e3b-a050-206d87063096 Running 9m47s v1.26.0
test-ci-cluster-md-0-d7bcbc5c5-txn9l test-ci-cluster test-ci-cluster-md-0-4dvz4 openstack:///52ae63f1-0bf2-4134-979b-0b87287ac162 Running 9m47s v1.26.0
test-ci-cluster-control-plane-hjvf7 test-ci-cluster test-ci-cluster-control-plane-tkx68 openstack:///6e2e4163-07c9-45da-8f70-c39773098214 Running 5m13s v1.26.0
test-ci-cluster-md-0-d7bcbc5c5-dtbwq test-ci-cluster test-ci-cluster-md-0-spldn openstack:///f779c4c2-1d43-4096-99e3-7bec4ec384ca Running 6m53s v1.26.0
test-ci-cluster-control-plane-xgw9x test-ci-cluster test-ci-cluster-control-plane-rngmn openstack:///43217072-3170-475f-89b7-49b22944b7d0 Deleting 33m v1.24.0

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)