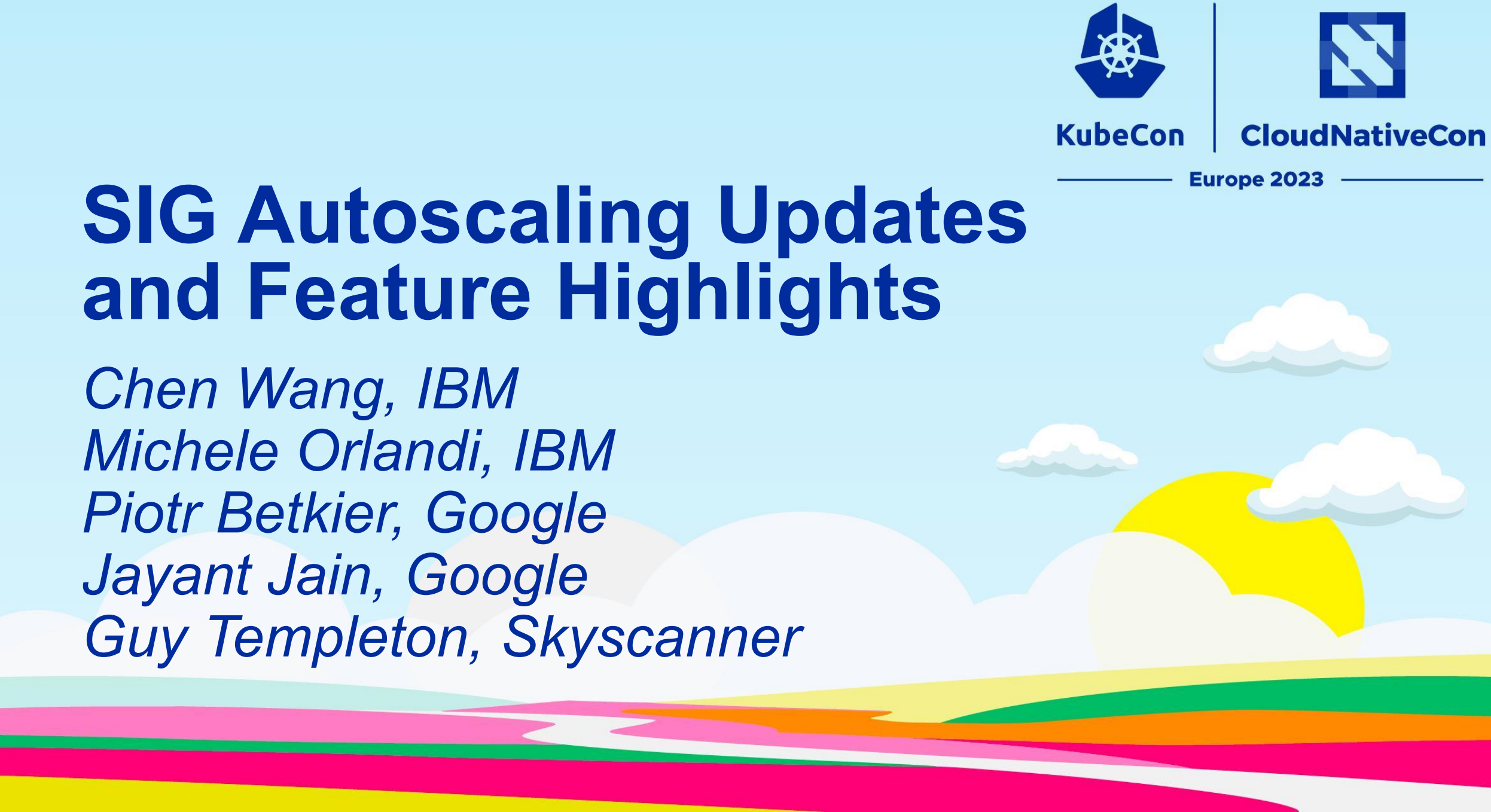

KubeCon
CloudNativeCon
Europe 2023
TiKV



SIG Autoscaling Updates and Feature Highlights

Chen Wang, IBM
Michele Orlandi, IBM
Piotr Betkier, Google
Jayant Jain, Google
Guy Templeton, Skyscanner

SIG Autoscaling

- Sub-Projects:
  - Balancer (New!)
  - Cluster Autoscaler
  - Horizontal Pod Autoscaling
  - Vertical Pod Autoscaling
  - Addon Resizer

SIG Autoscaling

- Other Updates:
  - HPA - Container Resource Scaling - being promoted to beta for 1.27
  - Cluster Autoscaler - Improved release process going forward
  - Vertical Pod Autoscaling - Gathering the work to enable taking advantage of dynamic pod resizing

SIG Autoscaling

- Where we need help:
  - Feedback
  - Issue Triage
  - PR Reviewers
  - Contributors

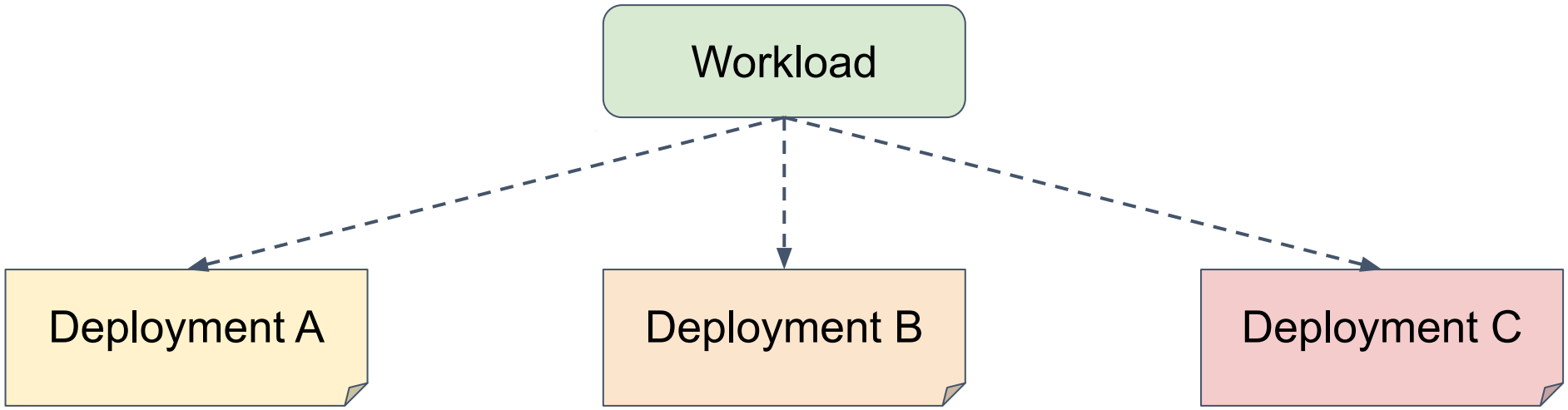
SIG Autoscaling

- How to get in touch:
  - github.com/kubernetes/autoscaler
  - SIG Meetings - Weekly at 16:00 CET
  - #sig-autoscaling

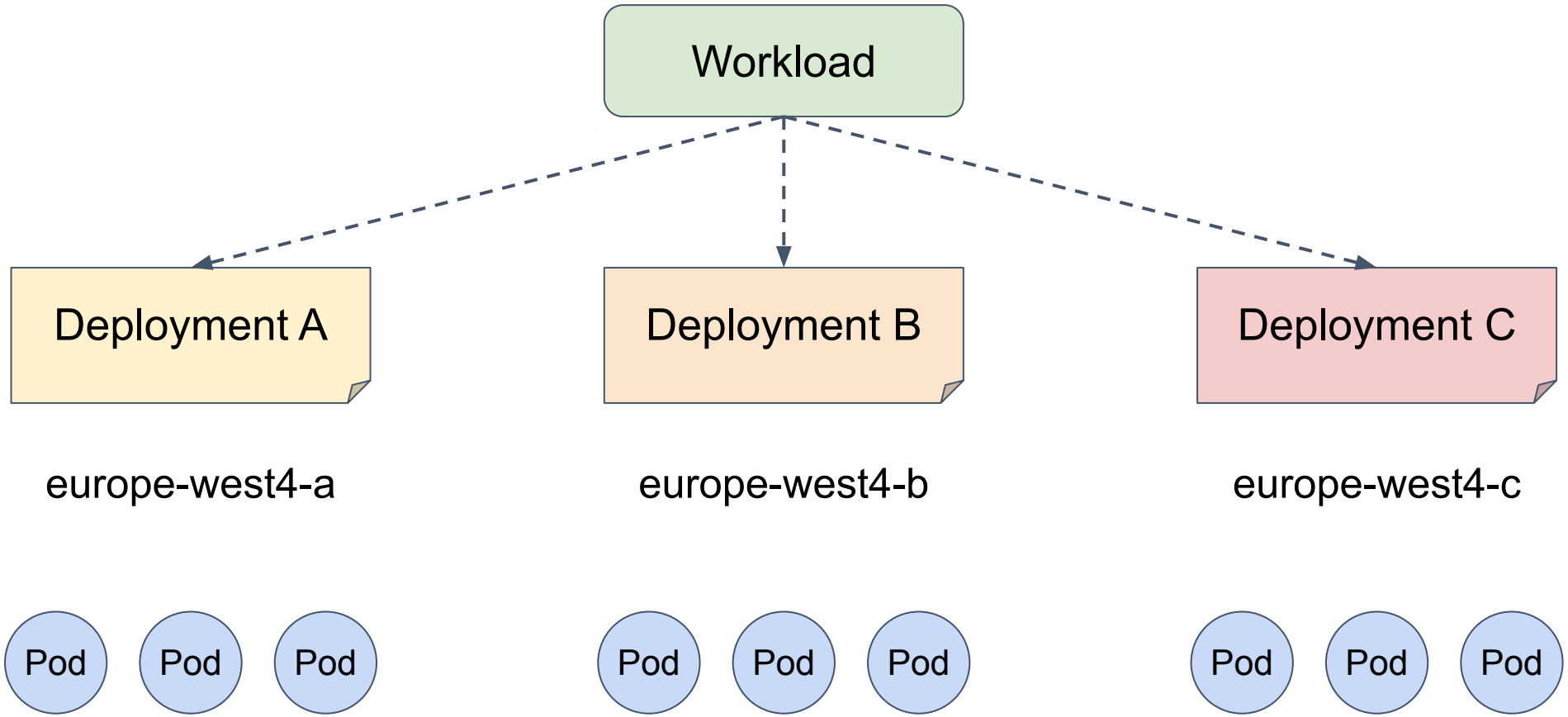
Balancer

- Balancer allows to
  - Balance Pod distribution across similar deployments
  - Autoscale Pods from similar deployments together
- New resource and its controller
  - Requires installing an optional component Balancer
  - Stage: alpha
  - See github.com/kubernetes/autoscaler

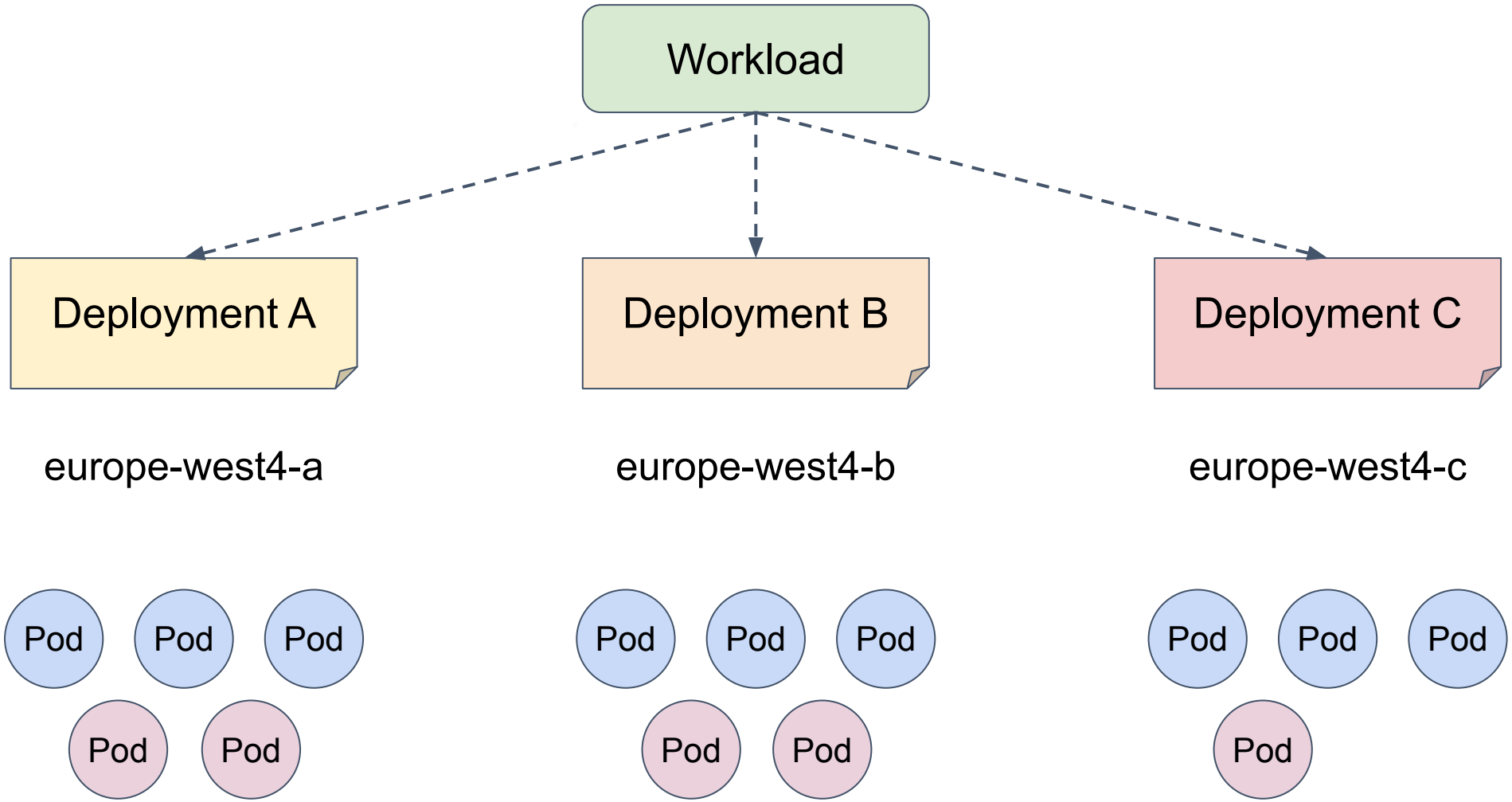
Balancer workload in multiple deployments



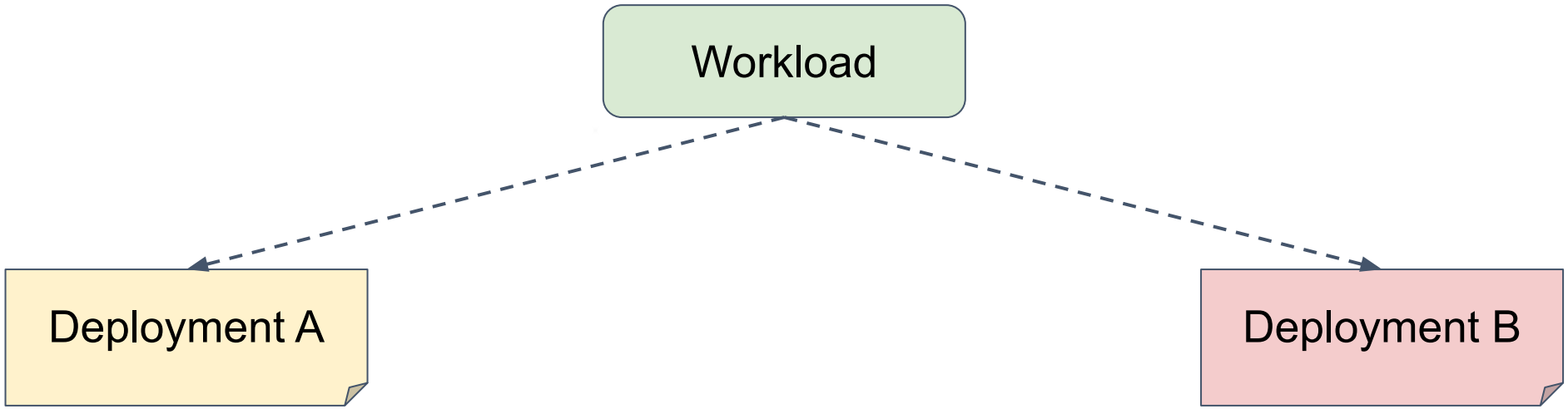
Balancer evenly distribute pods across zones



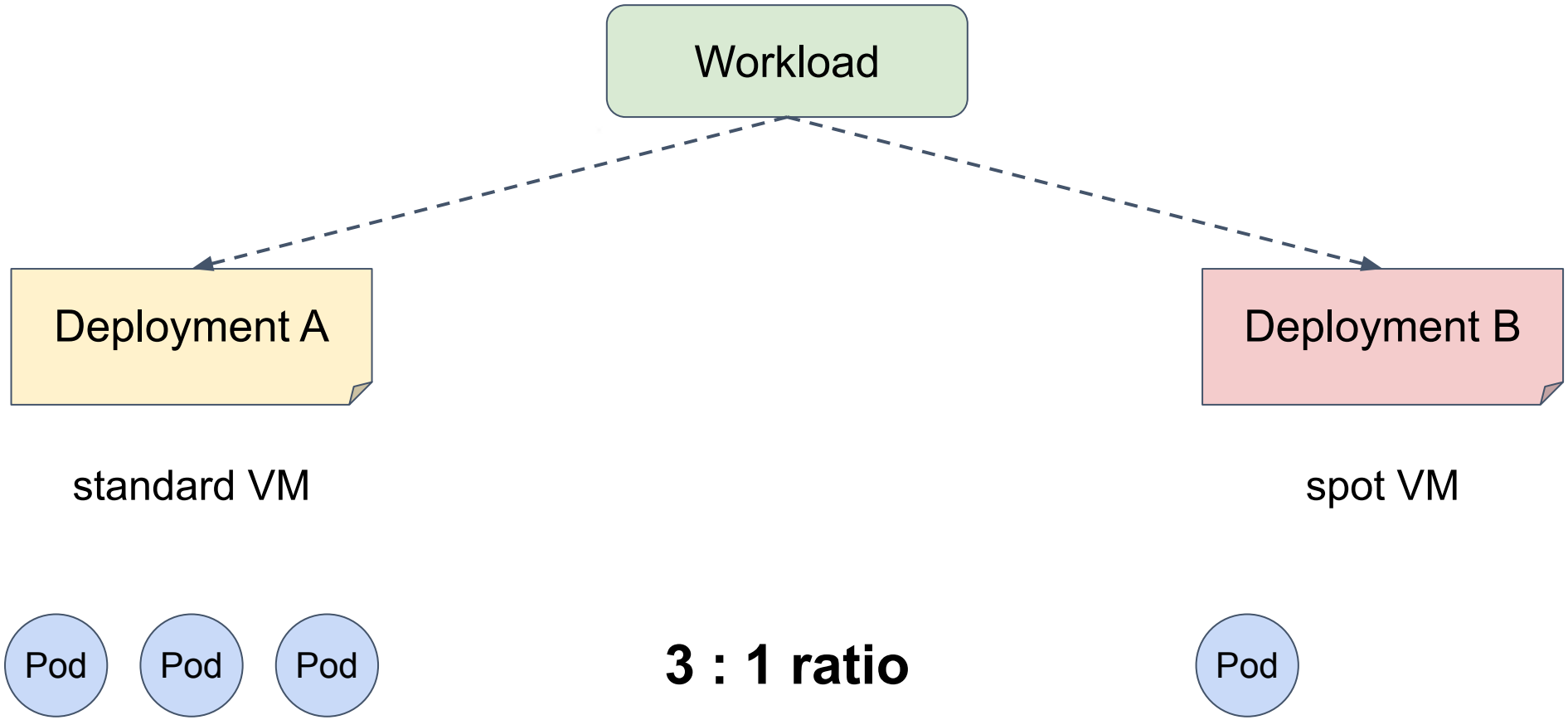
Balancer evenly distribute pods across zones



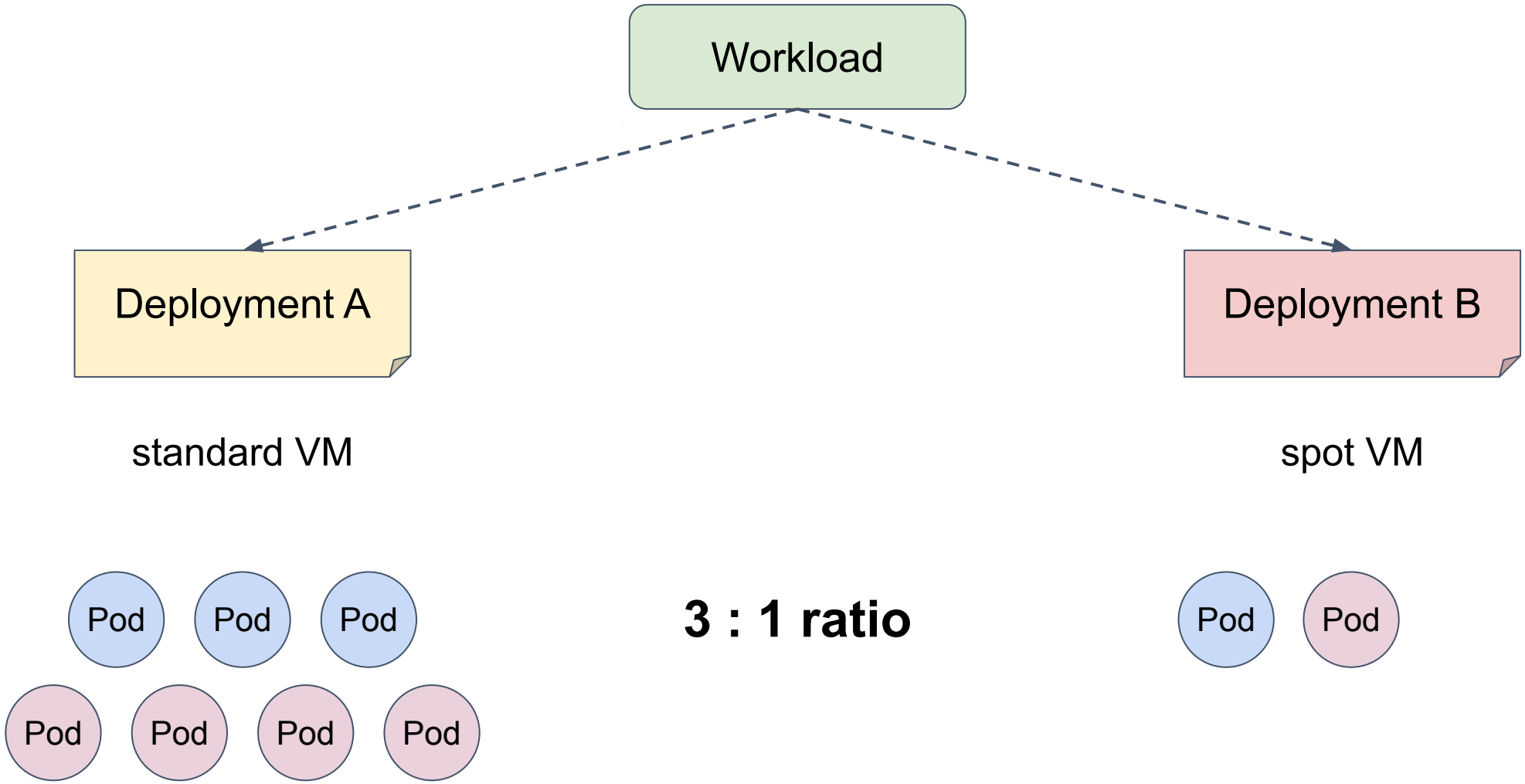
Balancer maintain pods ratio on spot VMs



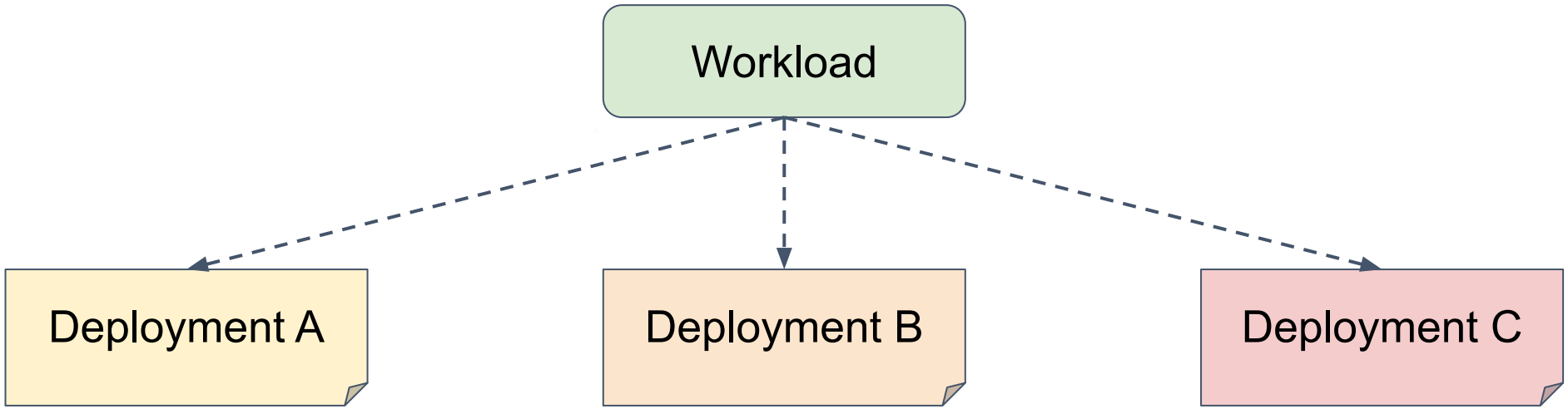
Balancer maintain pods ratio on spot VMs



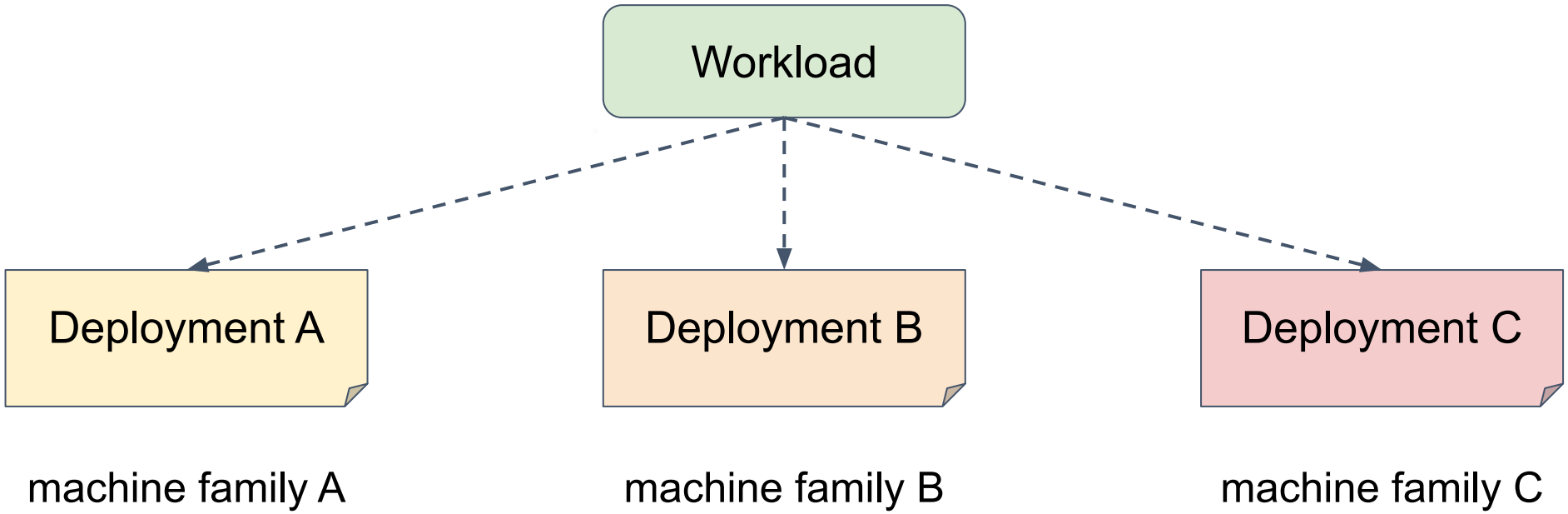
Balancer maintain pods ratio on spot VMs



Balancer workload on different machines



Balancer workload on different machines



Balancer how to use it

Balancer how to use it

I want to balance `myapp` between deployments `myapp-A` and `myapp-B` in 3:1 ratio.

Balancer how to use it

I want to balance `myapp` between deployments `myapp-A` and `myapp-B` in 3:1 ratio.

```yaml
apiVersion: balancer.x-k8s.io/v1alpha1
kind: Balancer
metadata:
  name: myapp-balancer
spec:
  selector:
    app: myapp
  targets:
    - name: myapp-A
      scaleTargetRef:
        apiVersion: apps/v1
        kind: Deployment
        name: myapp-A
      minReplicas: 1
      maxReplicas: 10
    - name: myapp-B
      scaleTargetRef:
        apiVersion: apps/v1
        kind: Deployment
        name: myapp-B
      minReplicas: 0
      maxReplicas: 10
  policy:
    type: proportional
    proportions:
      targetRatios: [myapp-A: 3, myapp-B: 1]
```

Balancer how to use it

I want to balance `myapp` between deployments `myapp-A` and `myapp-B` in 3:1 ratio.

```yaml
apiVersion: balancer.x-k8s.io/v1alpha1
kind: Balancer
metadata:
  name: myapp-balancer
spec:
  selector:
    app: myapp
  targets:
    - name: myapp-A
      scaleTargetRef:
        apiVersion: apps/v1
        kind: Deployment
        name: myapp-A
      minReplicas: 1
      maxReplicas: 10
    - name: myapp-B
      scaleTargetRef:
        apiVersion: apps/v1
        kind: Deployment
        name: myapp-B
      minReplicas: 0
      maxReplicas: 10
  policy:
    type: proportional
    proportions:
      targetRatios: [myapp-A: 3, myapp-B: 1]
```

Balancer how to use it

I want to balance `myapp` between deployments `myapp-A` and `myapp-B` in 3:1 ratio.

```yaml
apiVersion: balancer.x-k8s.io/v1alpha1
kind: Balancer
metadata:
  name: myapp-balancer
spec:
  selector:
    app: myapp
  targets:
    - name: myapp-A
      scaleTargetRef:
        apiVersion: apps/v1
        kind: Deployment
        name: myapp-A
      minReplicas: 1
      maxReplicas: 10
    - name: myapp-B
      scaleTargetRef:
        apiVersion: apps/v1
        kind: Deployment
        name: myapp-B
      minReplicas: 0
      maxReplicas: 10
  policy:
    type: proportional
    proportions:
      targetRatios: [myapp-A: 3, myapp-B: 1]
```

Balancer how to use it

I want to balance `myapp` between deployments `myapp-A` and `myapp-B` in 3:1 ratio.

With horizontal pod autoscaling

```yaml
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: myapp
spec:
  scaleTargetRef:
    apiVersion: balancer.x-k8s.io/v1beta1
    kind: Balancer
    name: myapp
  minReplicas: 1
  maxReplicas: 20
  metrics:
  - (...)
```

```yaml
apiVersion: balancer.x-k8s.io/v1alpha1
kind: Balancer
metadata:
  name: myapp-balancer
spec:
  selector:
    app: myapp
  targets:
    - name: myapp-A
      scaleTargetRef:
        apiVersion: apps/v1
        kind: Deployment
        name: myapp-A
      minReplicas: 1
      maxReplicas: 10
    - name: myapp-B
      scaleTargetRef:
        apiVersion: apps/v1
        kind: Deployment
        name: myapp-B
      minReplicas: 0
      maxReplicas: 10
  policy:
    type: proportional
    proportions:
      targetRatios: [myapp-A: 3, myapp-B: 1]
```

Balancer how to use it

Balancer how to use it

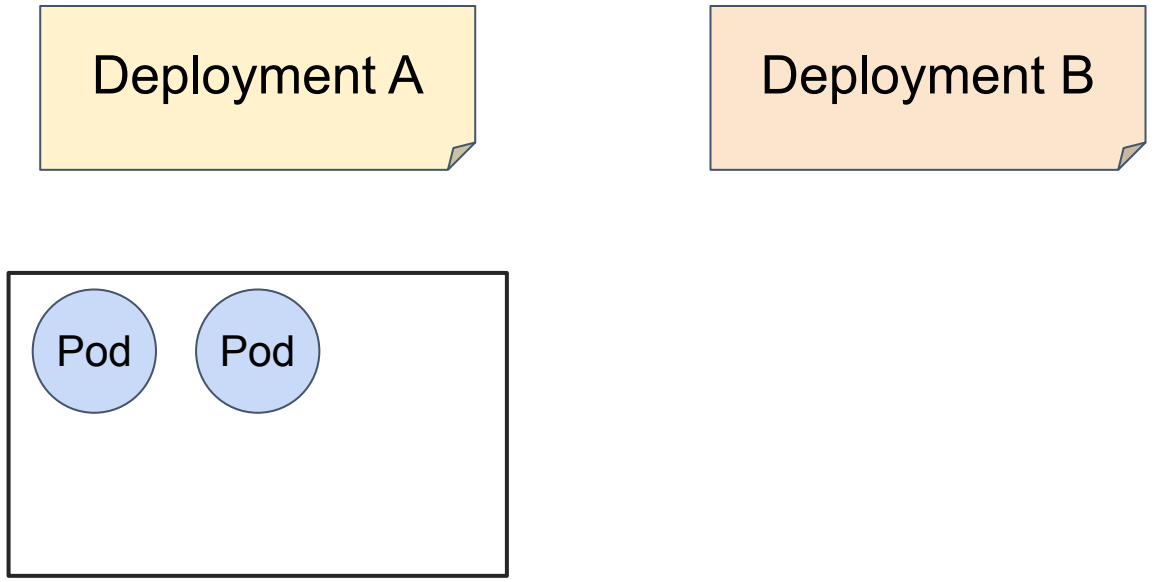
I want to run `myapp` in deployment `myapp-A`, but overflow to `myapp-B` if max replicas in `myapp-A` reached.

Deployment A

Deployment B

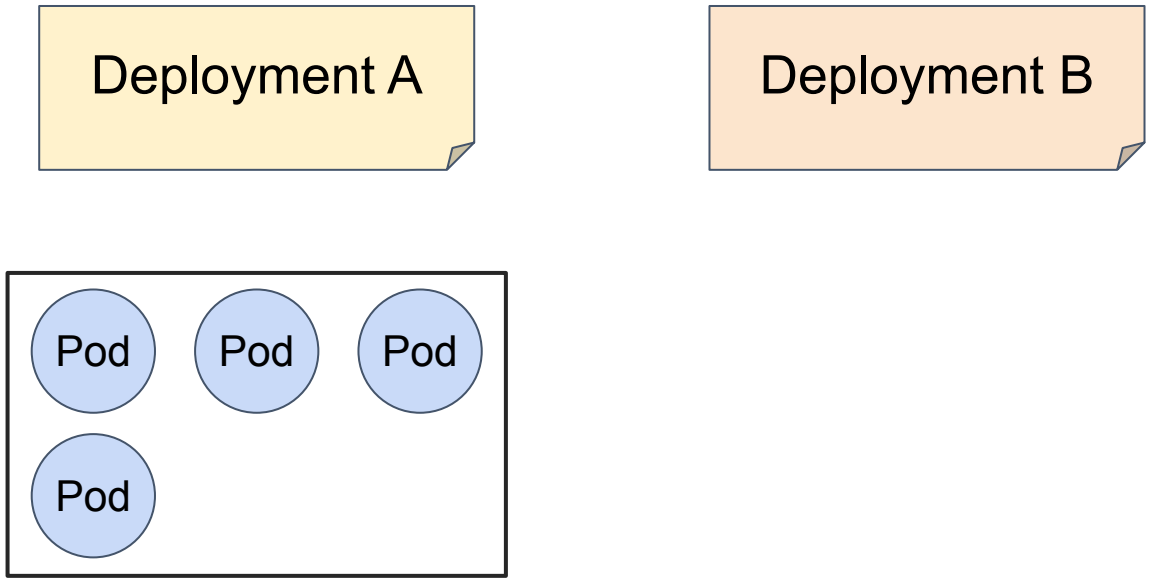
Balancer how to use it

I want to run `myapp` in deployment `myapp-A`, but overflow to `myapp-B` if max replicas in `myapp-A` reached.



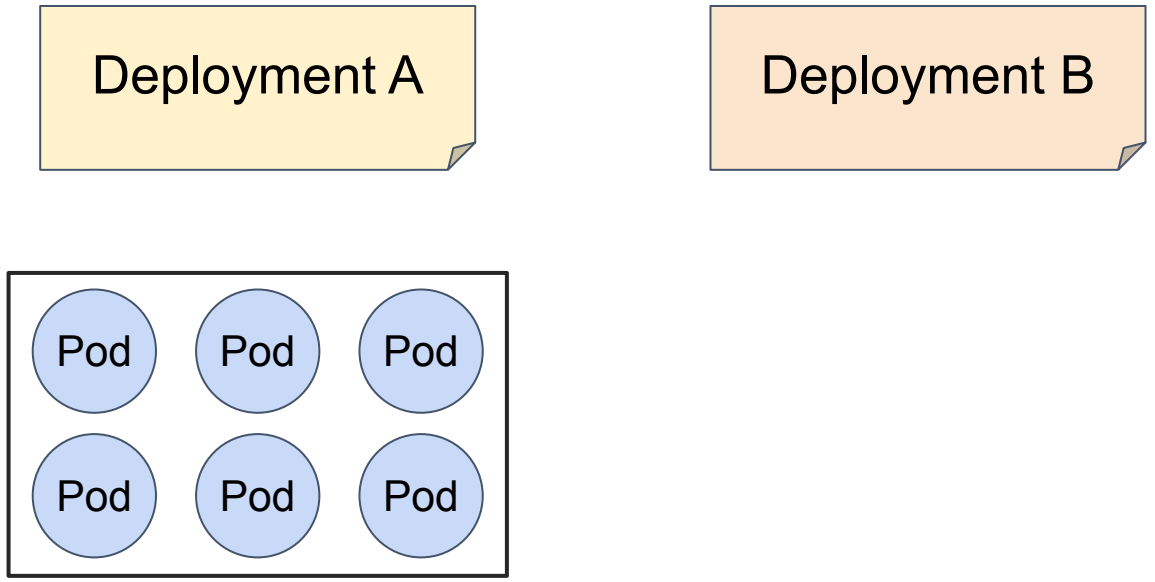
Balancer how to use it

I want to run `myapp` in deployment `myapp-A`, but overflow to `myapp-B` if max replicas in `myapp-A` reached.



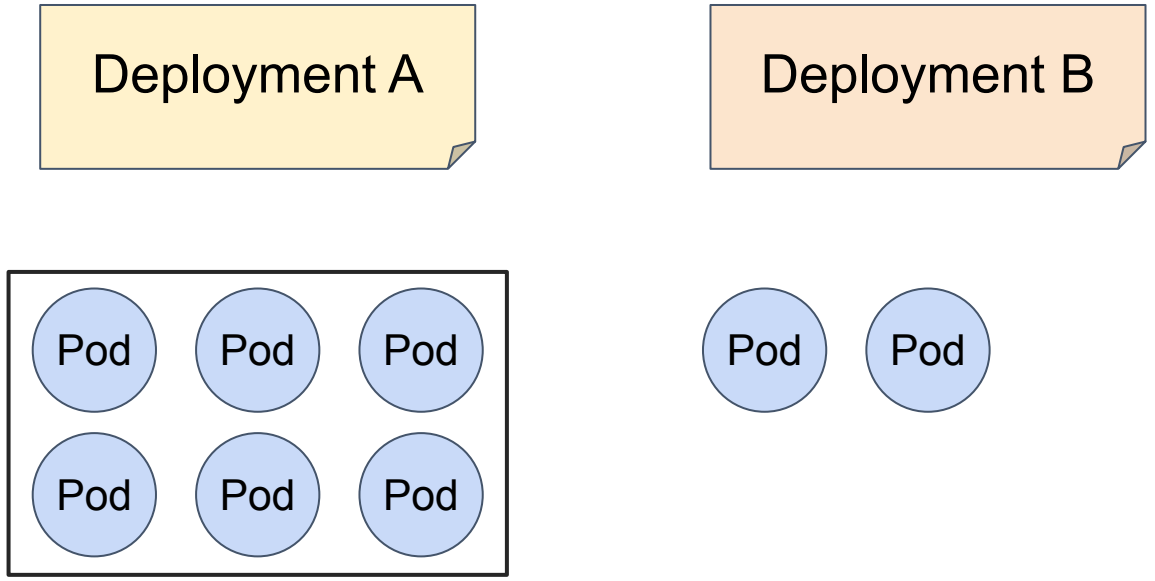
Balancer how to use it

I want to run `myapp` in deployment `myapp-A`, but overflow to `myapp-B` if max replicas in `myapp-A` reached.



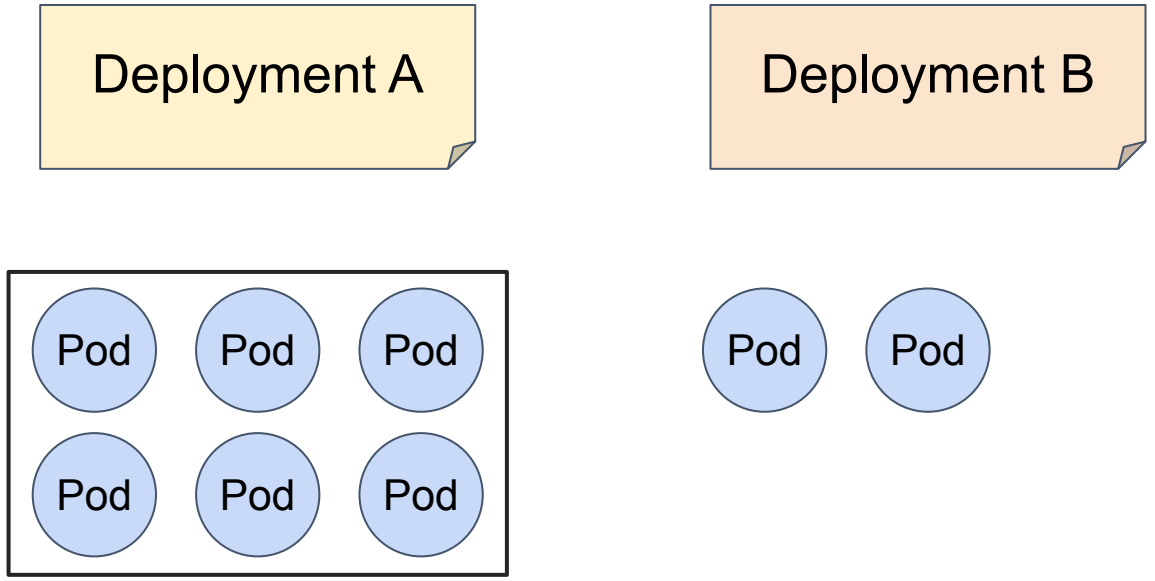
Balancer how to use it

I want to run `myapp` in deployment `myapp-A`, but overflow to `myapp-B` if max replicas in `myapp-A` reached.



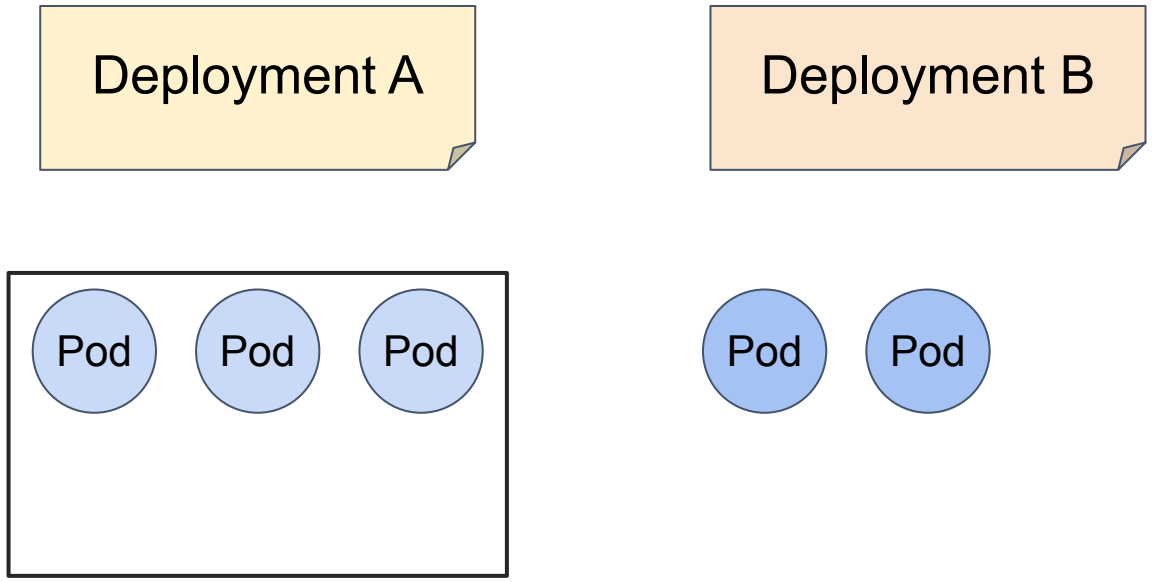
Balancer how to use it

I want to run `myapp` in deployment `myapp-A`, but overflow to `myapp-B` if max replicas in `myapp-A` reached.



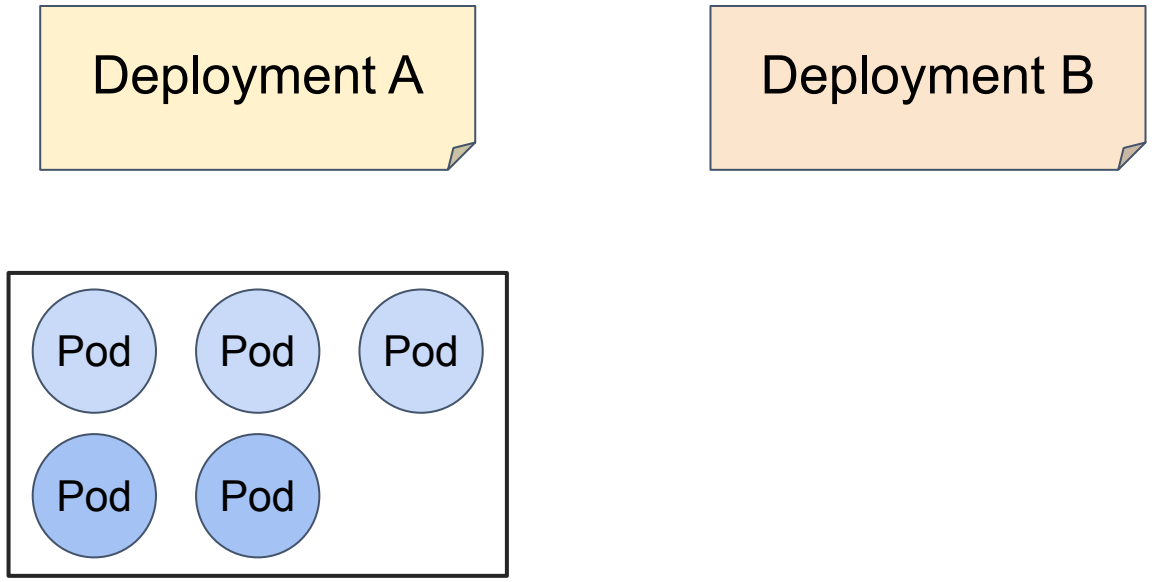
Balancer how to use it

I want to run `myapp` in deployment `myapp-A`, but overflow to `myapp-B` if max replicas in `myapp-A` reached.



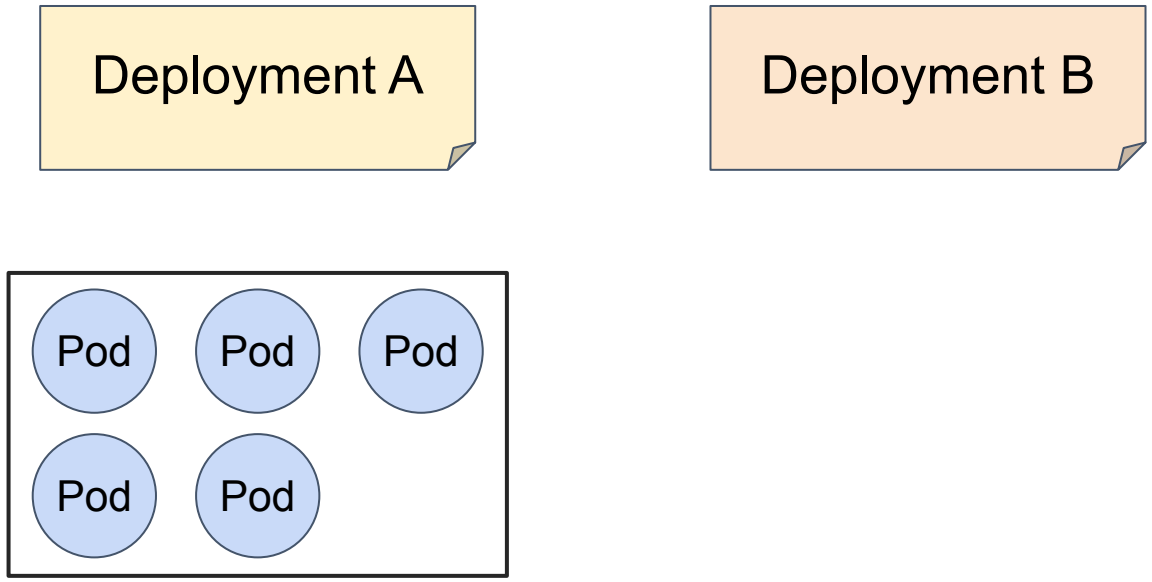
Balancer how to use it

I want to run `myapp` in deployment `myapp-A`, but overflow to `myapp-B` if max replicas in `myapp-A` reached.



Balancer how to use it

I want to run `myapp` in deployment `myapp-A`, but overflow to `myapp-B` if max replicas in `myapp-A` reached.

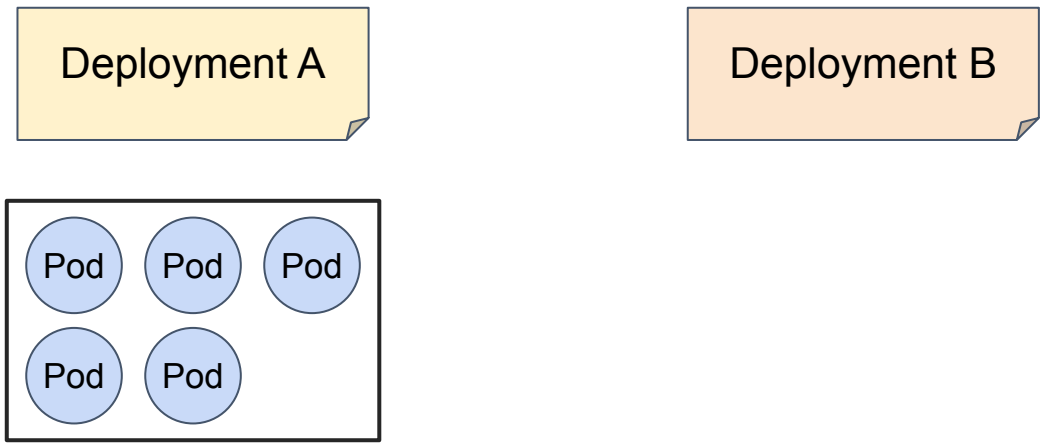



```yaml
apiVersion: balancer.x-k8s.io/v1alpha1
kind: Balancer
metadata:
  name: myapp-balancer
spec:
  selector:
    app: myapp
  targets:
    - name: myapp-A
      scaleTargetRef:
        apiVersion: apps/v1
        kind: Deployment
        name: myapp-A
      minReplicas: 1
      maxReplicas: 6
    - name: myapp-B
      scaleTargetRef:
        apiVersion: apps/v1
        kind: Deployment
        name: myapp-B
  policy:
    type: priority
    priority:
      targetOrder: [myapp-A, myapp-B]
```

Balancer how to use it

I want to run `myapp` in deployment `myapp-A`, but overflow to `myapp-B` if max replicas in `myapp-A` reached.

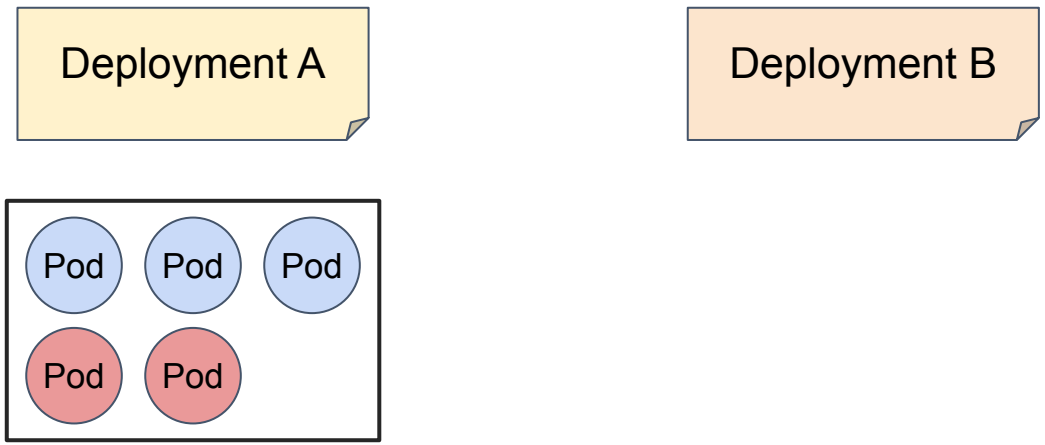
Also fallback to `myapp-B` if cannot start Pods in `myapp-A`.



Balancer how to use it

I want to run `myapp` in deployment `myapp-A`, but overflow to `myapp-B` if max replicas in `myapp-A` reached.

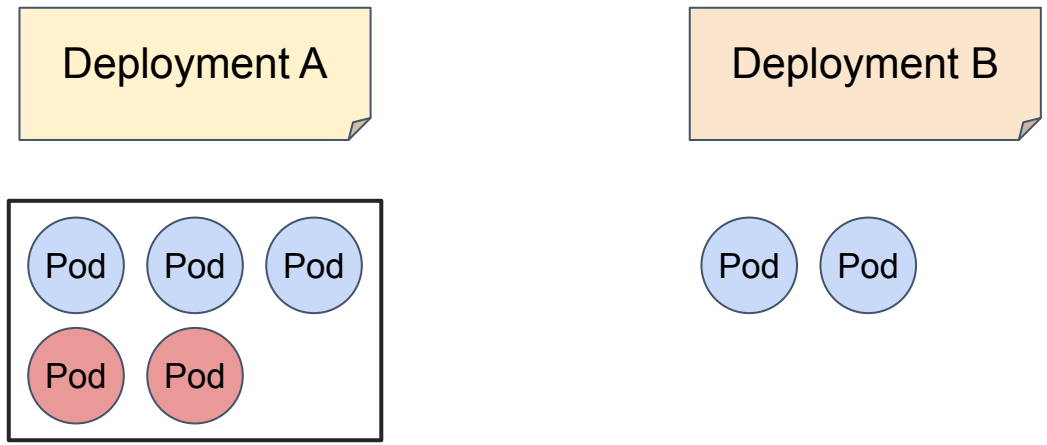
Also fallback to `myapp-B` if cannot start Pods in `myapp-A`.



Balancer how to use it

I want to run `myapp` in deployment `myapp-A`, but overflow to `myapp-B` if max replicas in `myapp-A` reached.

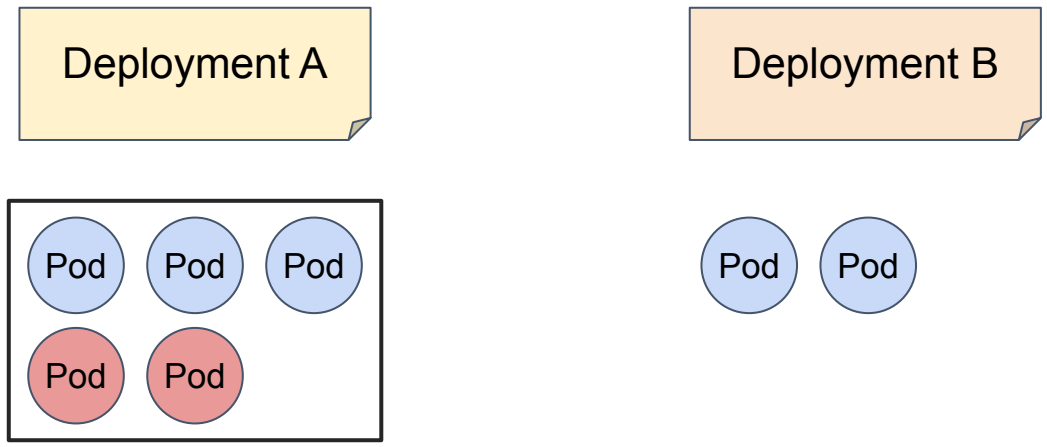
Also fallback to `myapp-B` if cannot start Pods in `myapp-A`.



Balancer how to use it

I want to run `myapp` in deployment `myapp-A`, but overflow to `myapp-B` if max replicas in `myapp-A` reached.

Also fallback to `myapp-B` if cannot start Pods in `myapp-A`.

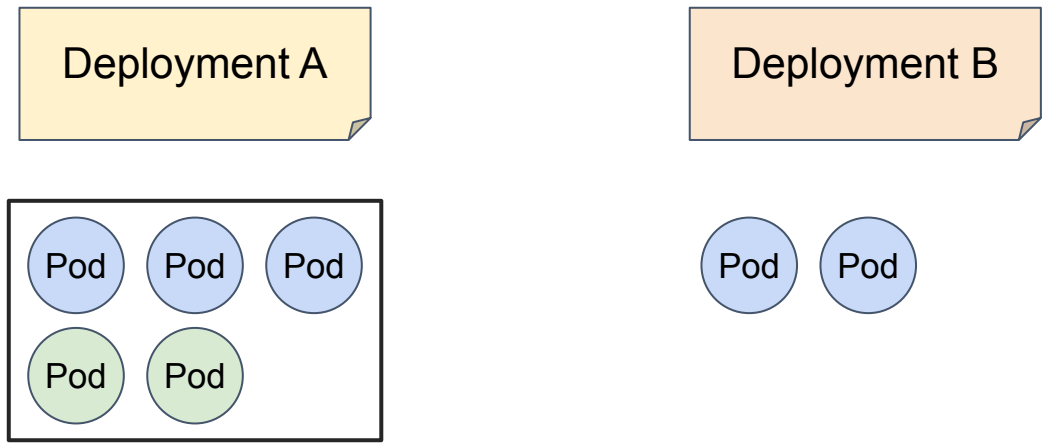



```yaml
apiVersion: balancer.x-k8s.io/v1alpha1
kind: Balancer
metadata:
  name: myapp-balancer
spec:
  selector:
    app: myapp
  targets:
    - name: myapp-A
      scaleTargetRef:
        apiVersion: apps/v1
        kind: Deployment
        name: myapp-A
      minReplicas: 1
      maxReplicas: 10
    - name: myapp-B
      scaleTargetRef:
        apiVersion: apps/v1
        kind: Deployment
        name: myapp-B
  policy:
    type: priority
    priority:
      targetOrder: [myapp-A, myapp-B]
    fallback:
      enabled: true
      startupTimeout: 5min
```

Balancer how to use it

I want to run `myapp` in deployment `myapp-A`, but overflow to `myapp-B` if max replicas in `myapp-A` reached.

Also fallback to `myapp-B` if cannot start Pods in `myapp-A`.

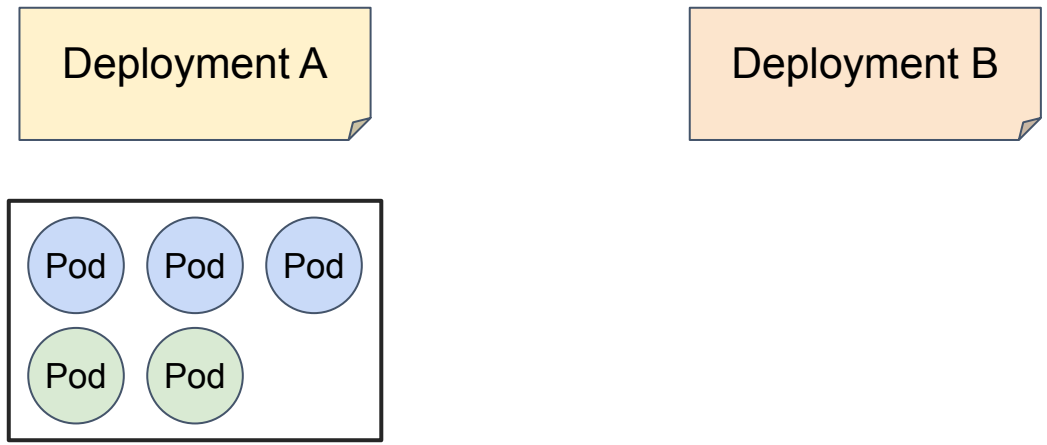



```yaml
apiVersion: balancer.x-k8s.io/v1alpha1
kind: Balancer
metadata:
  name: myapp-balancer
spec:
  selector:
    app: myapp
  targets:
    - name: myapp-A
      scaleTargetRef:
        apiVersion: apps/v1
        kind: Deployment
        name: myapp-A
      minReplicas: 1
      maxReplicas: 10
    - name: myapp-B
      scaleTargetRef:
        apiVersion: apps/v1
        kind: Deployment
        name: myapp-B
  policy:
    type: priority
    priority:
      targetOrder: [myapp-A, myapp-B]
    fallback:
      enabled: true
      startupTimeout: 5min
```

Balancer how to use it

I want to run `myapp` in deployment `myapp-A`, but overflow to `myapp-B` if max replicas in `myapp-A` reached.

Also fallback to `myapp-B` if cannot start Pods in `myapp-A`.




```yaml
apiVersion: balancer.x-k8s.io/v1alpha1
kind: Balancer
metadata:
  name: myapp-balancer
spec:
  selector:
    app: myapp
  targets:
    - name: myapp-A
      scaleTargetRef:
        apiVersion: apps/v1
        kind: Deployment
        name: myapp-A
      minReplicas: 1
      maxReplicas: 10
    - name: myapp-B
      scaleTargetRef:
        apiVersion: apps/v1
        kind: Deployment
        name: myapp-B
  policy:
    type: priority
    priority:
      targetOrder: [myapp-A, myapp-B]
    fallback:
      enabled: true
      startupTimeout: 5min
```

Balancer summary

- Use Balancer to
  - Balance Pod distribution across similar deployments
  - Autoscale Pods from similar deployments together
- Learn more at github.com/kubernetes/autoscaler
- Give feedback on GitHub issues!

Debugging Snapshotter

What are we trying to solve?

- CA logs information about the decisions taken, but we don't log what data these decisions are based on as CA internally simulates the behaviour of the entire cluster when making decisions. When something goes wrong, we usually need to understand how these decisions were taken.
- RCA of the issue usually takes time, and the internal state of CA changes when we mitigate the issue, thereby making it more difficult to debug the issue.

Debugging Snapshotter:
It's a tool to visualize the internal state of cluster-autoscaler at a point in time to help debug autoscaling issues.

Debugging Snapshotter

Common use-cases

- Scale up and scale down not working, with special focus on scale from zero nodes.
- Mismatch between scheduler and cluster-autoscaler decisions.
- Mismatch in resource availability on the node.

Debugging Snapshotter

KubeCon CloudNativeCon Europe 2023

What data is captured?

```
// DebuggingSnapshotImpl is the struct used to collect all the data to be output.
// Please add all new output fields in this struct. This is to make the data
// encoding/decoding easier as the single object going into the decoder
type DebuggingSnapshotImpl struct {
        NodeList                     []*ClusterNode          `json:"NodeList"`
        UnscheduledPodsCanBeScheduled []*v1.Pod              `json:"UnscheduledPodsCanBeScheduled"`
        Error                        string                  `json:"Error,omitempty"`
        StartTimestamp               time.Time               `json:"StartTimestamp"`
        EndTimestamp                 time.Time               `json:"EndTimestamp"`
        TemplateNodes                map[string]*ClusterNode `json:"TemplateNodes"`
}
```

https://github.com/kubernetes/autoscaler/blob/master/cluster-autoscaler/debuggingsnapshot/debugging_snapshot.go#L63

Debugging Snapshotter

How does it work?

- Captures a set of internal fields (when it receives a request) and returns a well-formed JSON.
- After receiving the request, in the following cluster-autoscaler loop, snapshotter will snapshot the current state, limited to a single loop (no cross-loop data).
- An HTTP request blocks for the duration of the snapshot generation, and returns the JSON as a HTTP response to the request.
- SSH to server running the leader cluster-autoscaler, and curl `http://127.0.0.1:8085/snapshotz`

Debugging Snapshotter

Demo

Debugging Snapshotter quick-start

- Enable the snapshotter by adding the following flag to cluster-autoscaler manifest
  `--debugging-snapshot-enabled=true`
- Available in cluster-autoscaler versions *1.24+*
- Detailed instructions in cluster-autoscaler FAQs.
  https://github.com/kubernetes/autoscaler/blob/master/cluster-autoscaler/FAQ.md#how-to
- Looking forward to hearing feedback! (and possible extensions to make life easier)

Session QR Codes will be
sent via email before the event

Please scan the QR Code above
to leave feedback on this session