


KubeCon | CloudNativeCon
Europe 2022

SIG Autoscaling

Updates and Feature Highlights

Joachim Bartosik
Software Engineer
Google

Michael McCune
Principal Software Engineer
Red Hat

David Morrison
Staff Software Engineer
Airbnb

Guy Templeton
Principal Software Engineer
Skyscanner

Forecast

- Introductions
- Horizontal Pod Autoscaler
  - API v2
- Cluster Autoscaler
  - gRPC custom expander interface
- Vertical Pod Autoscaler
  - Support for alternative recommenders
  - Support for updating controllers with only 1 pod
  - Expect releases with each K8s release in the future
- SIG Autoscaling Community

Introductions

- Michael McCune
- David Morrison
- Joachim Bartosik
- Guy Templeton


Horizontal Pod Autoscaler

HPA API v2 Status

- v2 API is stable
  - from Kubernetes 1.23
- v2beta2 API is deprecated
  - must be served in Kubernetes 1.23, 1.24, 1.25
  - will be removed in Kubernetes 1.26
- v2beta1 API is deprecated as of May 2018
  - will be removed in Kubernetes 1.25

HPA API v2 Changes

- No changes to serialized fields
- Minor changes to internal Go API
  - `MaxPolicySelect` -> `MaxChangePolicySelect`
- See [GitHub Kubernetes/Kubernetes pull request #102534]


KubeCon | CloudNativeCon

Europe 2022

Cluster Autoscaler
Custom Expander Interface

What is an expander?

```
func ScaleUp(...) {
  podEquivalenceGroups := buildPodEquivalenceGroups(unschedulablePods)

  for _, nodeGroup := range nodeGroups {
    option, err := computeExpansionOption(
       context, podEquivalenceGroups, nodeGroup, nodeInfo, upcomingNodes)
  }

  bestOption := context.ExpanderStrategy.BestOption(options, nodeInfos)

  if bestOption != nil && bestOption.NodeCount > 0 {
    // Scale up the chosen node group
  }
}
```

What is an expander?

```
func ScaleUp(...) {
  podEquivalenceGroups := buildPodEquivalenceGroups(unschedulablePods)

  for _, nodeGroup := range nodeGroups {
    option, err := computeExpansionOption(
       context, podEquivalenceGroups, nodeGroup, nodeInfo, upcomingNodes)
  }

  bestOption := context.ExpanderStrategy.BestOption(options, nodeInfos)

  if bestOption != nil && bestOption.NodeCount > 0 {
    // Scale up the chosen node group
  }
}
```

Types of Expanders (source)

- random (default) - does what it says on the tin
- most pods - picks the node group that would be able to accommodate the largest number of pods on scale-up
- least waste - picks the node group that has the least unused CPU/memory after scaling up
- price - picks the node group which costs the least (GKE only)
- priority - picks a node group according to a user-specified priority ladder
- custom gRPC expander (this is new!) - allows for arbitrary, user-specified expansion behaviour

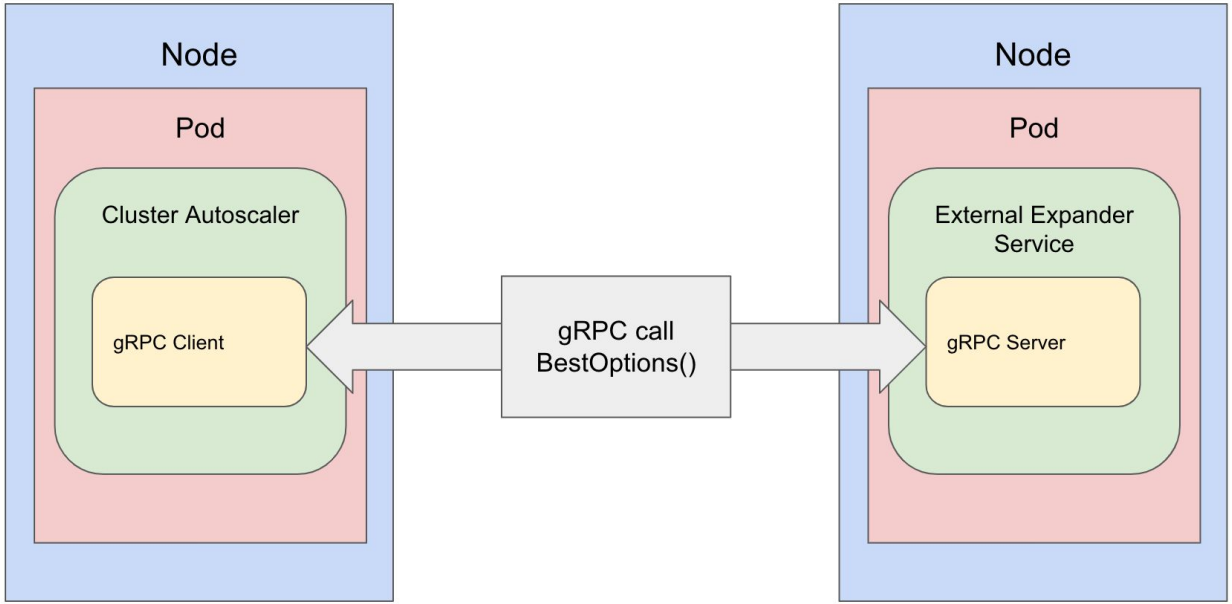
Types of Expanders (source)

- random (default) - does what it says on the tin
- most pods - picks the node group that would be able to accommodate the largest number of pods on scale-up
- least waste - picks the node group that has the least unused CPU/memory after scaling up
- price - picks the node group which costs the least (GKE only)
- priority - picks a node group according to a user-specified priority ladder
- **custom gRPC expander (this is new!) - allows for arbitrary, user-specified expansion behaviour**

Why a custom expander type?

- Allows for custom (business-specific) scale-up logic to be specified
  - Example: selecting different spot markets depending on time of day
- Expander logic can be developed out-of-band with Cluster Autoscaler releases, providing more flexibility

Expander Design



Expander Interface

```
service Expander {
  rpc BestOptions (BestOptionsRequest)
    returns (BestOptionsResponse) {}
}

message Option {
  // only need the ID of node to uniquely
  // identify the nodeGroup, used in the
  // nodeInfo map.
  string nodeGroupId = 1;
  int32 nodeCount = 2;
  string debug = 3;
  repeated Pod pod = 4;
}
```

```
message BestOptionsRequest {
  repeated Option options = 1;
  // key is node id from options
  map<string, Node> nodeInfoMap = 2;
}

message BestOptionsResponse {
  repeated Option options = 1;
}
```

Example Expander Code

```
func Serve(certPath string, keyPath string, port uint) {
     var grpcServer *grpc.Server
     // set up server here

     expanderServerImpl := NewExpanderServerImpl()
     protos.RegisterExpanderServer(grpcServer, expanderServerImpl)

     if err := grpcServer.Serve(netListener); err != nil {
          log.Fatalf("failed to serve: %s", err)
     }
}
```

Example Expander Code

```
func BestOptions(req *protos.BestOptionsRequest) (*protos.BestOptionsResponse, error) {
    longest := 0
    var choice *protos.Option
    for _, opt := range req.GetOptions() {
        if len(opt.NodeGroupId) > longest {
            choice = opt
        }
    }
    return &protos.BestOptionsResponse{Options: []*protos.Option{choice}}, nil
}
```

How to configure the custom expander?

```
./cluster-autoscaler \
  --expander=grpc,priority \
  --grpc-expander-url=ca-grpc-expander.svc.cluster.local:12345 \
  --grpc-expander-cert=/etc/ssl/certs/ca-grpc-expander.crt
```

For More Info

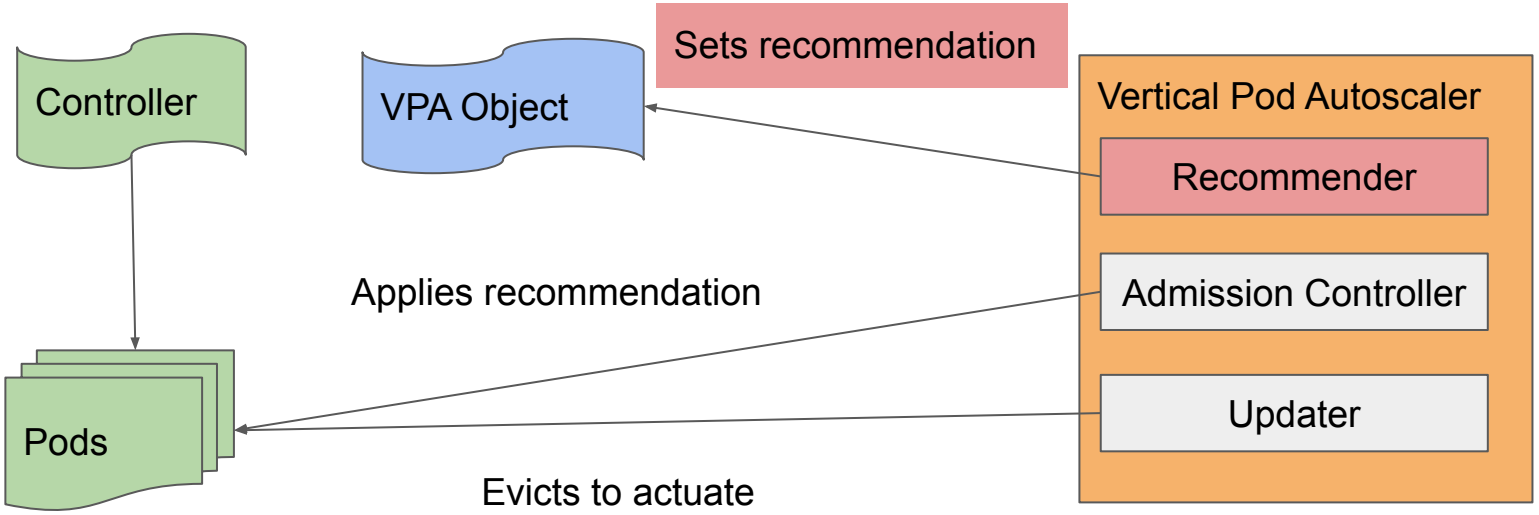
- Design Proposal
- Pull Request
- gRPC Expander README
- gRPC Expander Example Code
- Dynamic Kubernetes Cluster Scaling at Airbnb


Vertical Pod Autoscaler Updates

Overview

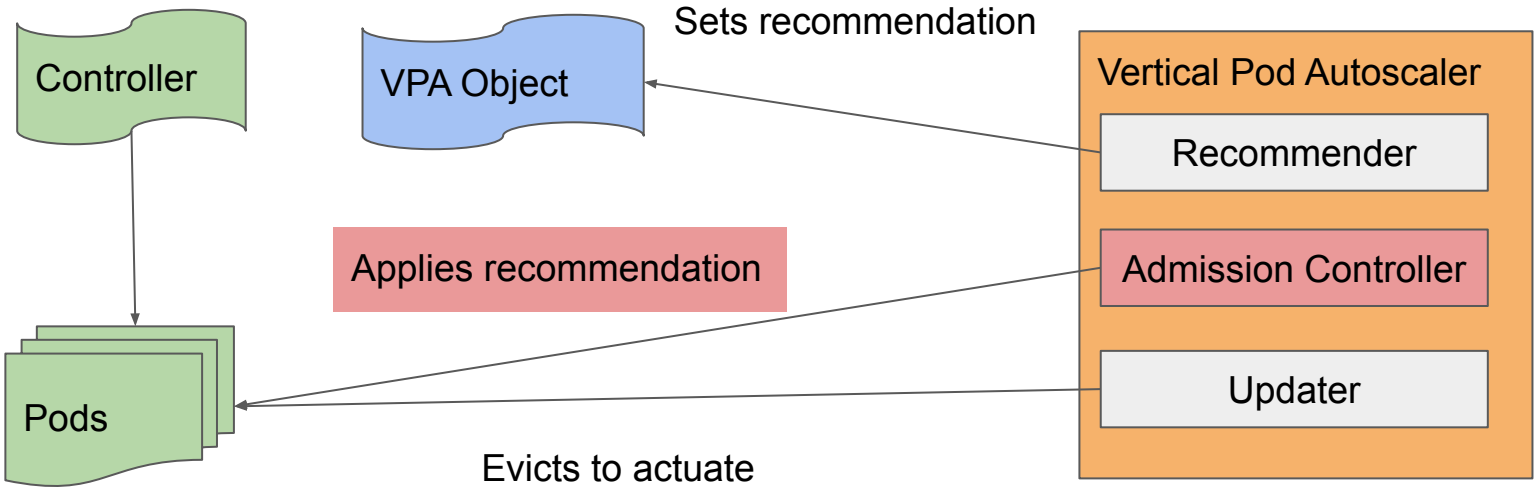
- What does Vertical Pod Autoscaler do;
- Enhancement: Alternative recommender support;
- Enhancement: Per VPA object min replicas;
- Releases.

What Vertical Pod Autoscaler does



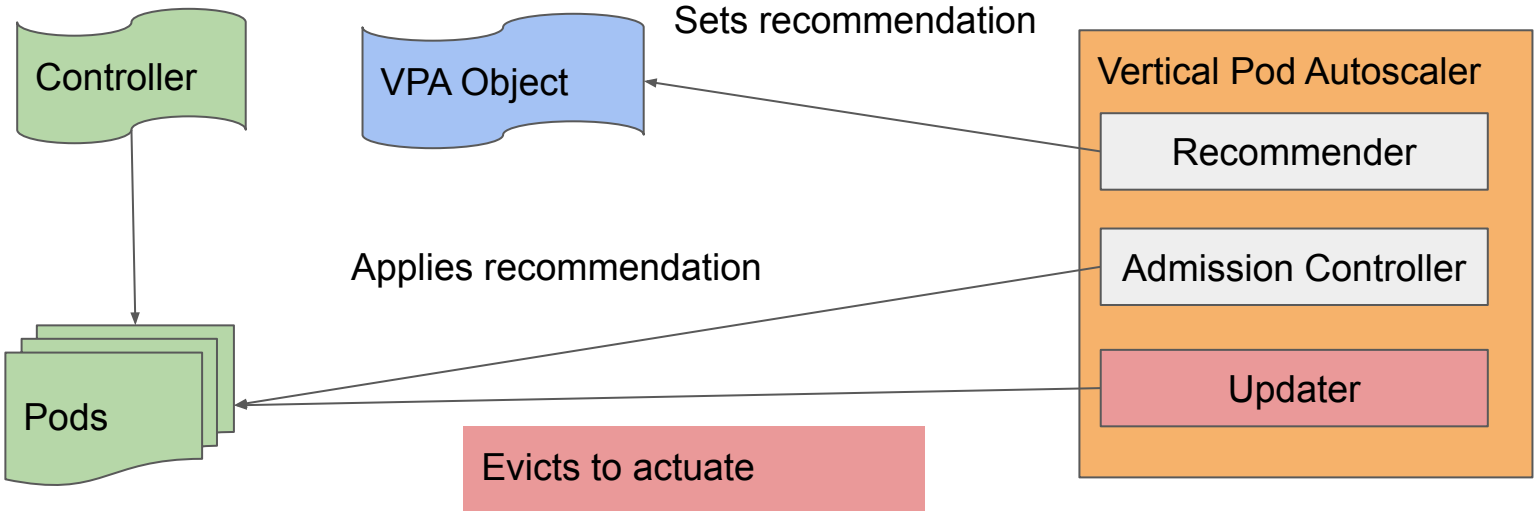
All modes: Off, Initial, and Auto / Recreate.

What Vertical Pod Autoscaler does



Modes: Initial, and Auto / Recreate.

What Vertical Pod Autoscaler does



Only in Auto / Recreate mode.

Alternative Recommender Support - Why

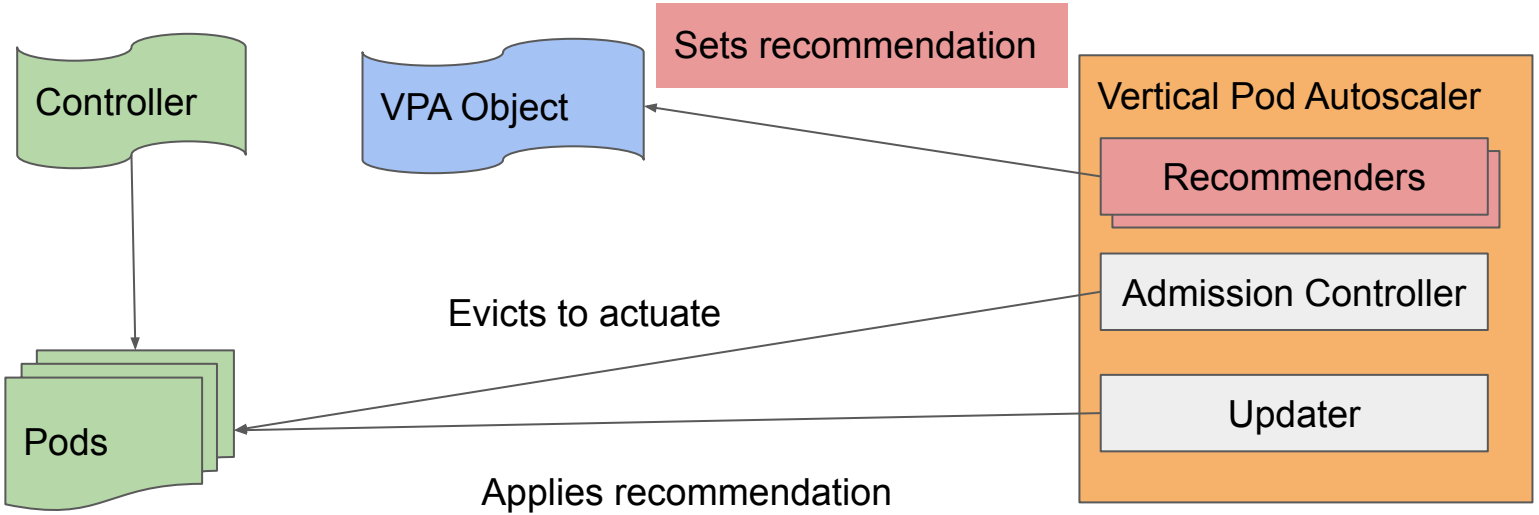
Different usage patterns:

- Weekly vs longer;
- Making window longer slows down reaction.

Different requirements:

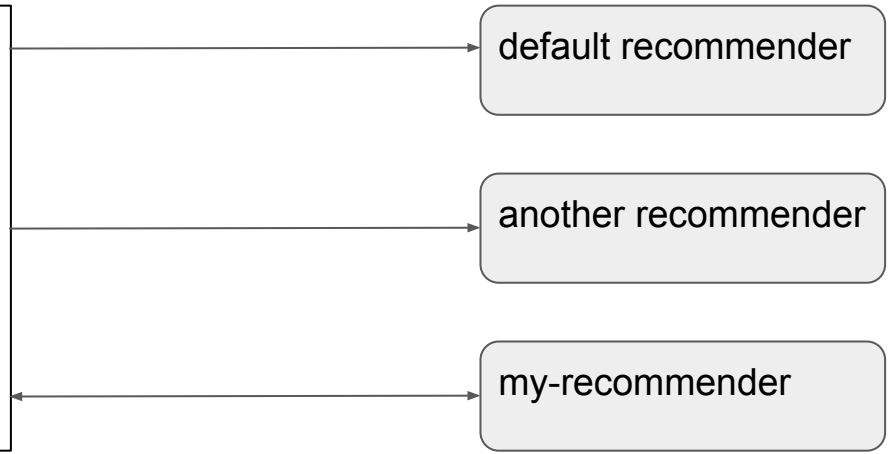
- Need to process load increase when it happens vs can take a while.

Alternative Recommender Support - what



Alternative Recommender Support - usage

```
apiVersion: autoscaling.k8s.io/v1
kind: VerticalPodAutoscaler
  ...
  spec:
  recommenders:
  - name: my-recommender
```



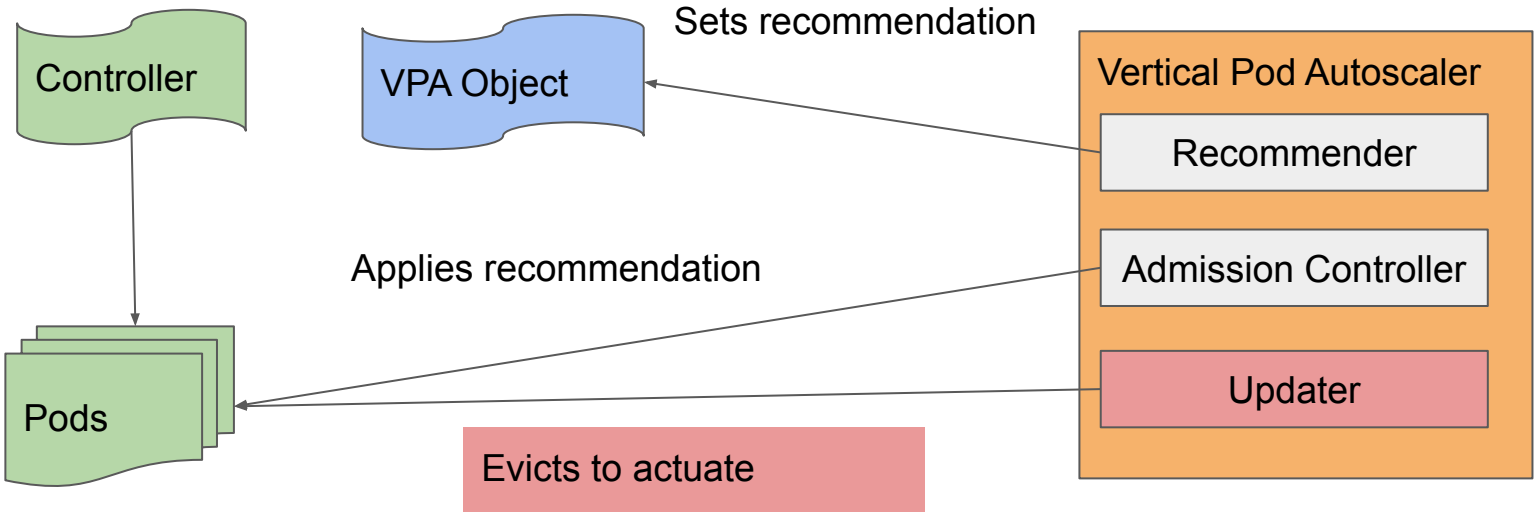
Setting up recommenders

- You have to implement your own
  - For now (hopefully)
- Only one recommender can write to a VPA object
  - Otherwise recommendations will flap
- Recommender can recognize more than one name
- Default recommender writes recommendation:
  - When no recommender is specified (for backward compatibility)
  - When “default” recommender is explicitly specified

Per VPA object MinReplicas - what & why

- It’s about evicting pods to apply a new recommendation
- By default VPA Updater won’t evict pod if it’s the only one running in its controller
  - Because that will definitely disrupt operations while the pod restarts
- This is a problem if expect to have only one pod in a controller
- There was `--min-replicas` flag but:
  - It changes behavior for all controllers (and possibly you might want different behavior for different controllers)
  - It doesn’t work if you don’t manage the cluster

Per VPA object MinReplicas - Where



Per VPA object MinReplicas - how

```
apiVersion:
autoscaling.k8s.io/v1
kind:
VerticalPodAutoscaler
  name: vpa2
  updatePolicy:
    minReplicas: 1
```

Releases

- Ad-hoc before
  - 0.9.2 on 2021-01-18
  - 0.10.0 on 2022-01-26
- We want to do them when new K8s release happens (so 3 / year)

Learn More

- KEP: Support Customized Recommenders for Vertical Pod Autoscalers
- KEP: MinReplicas per VPA object
- Ideas for the future:
  - Recommender-specific params
  - Multiple recommendations visible in one VPA object to make comparison and choice easier
  - Making default recommender more flexible so you can tweak its params and run multiple instances with different params under different names



KubeCon | CloudNativeCon
Europe 2022

SIG Autoscaling Community

The SIG Wants Your Help

We own a lot for the number of maintainers we have, this also means we have lots of opportunities!

- Feature requests and implementation
- Bug triage/response
- Infrastructure Improvements

The SIG Wants Your Help

- Expand our maintainers
- Improve extensibility of our owned subprojects

Thanks!

Community Charter

- https://github.com/kubernetes/community/tree/master/sig-autoscaling

Mailing List

- https://groups.google.com/g/kubernetes-sig-autoscaling

Office Hours

- https://docs.google.com/document/d/1RvhQAEIrVLHbyNnuaT99-6u9ZUMp7BfkPupT2LAZK7w

Join us on Kubernetes Slack in the #sig-autoscaling channel