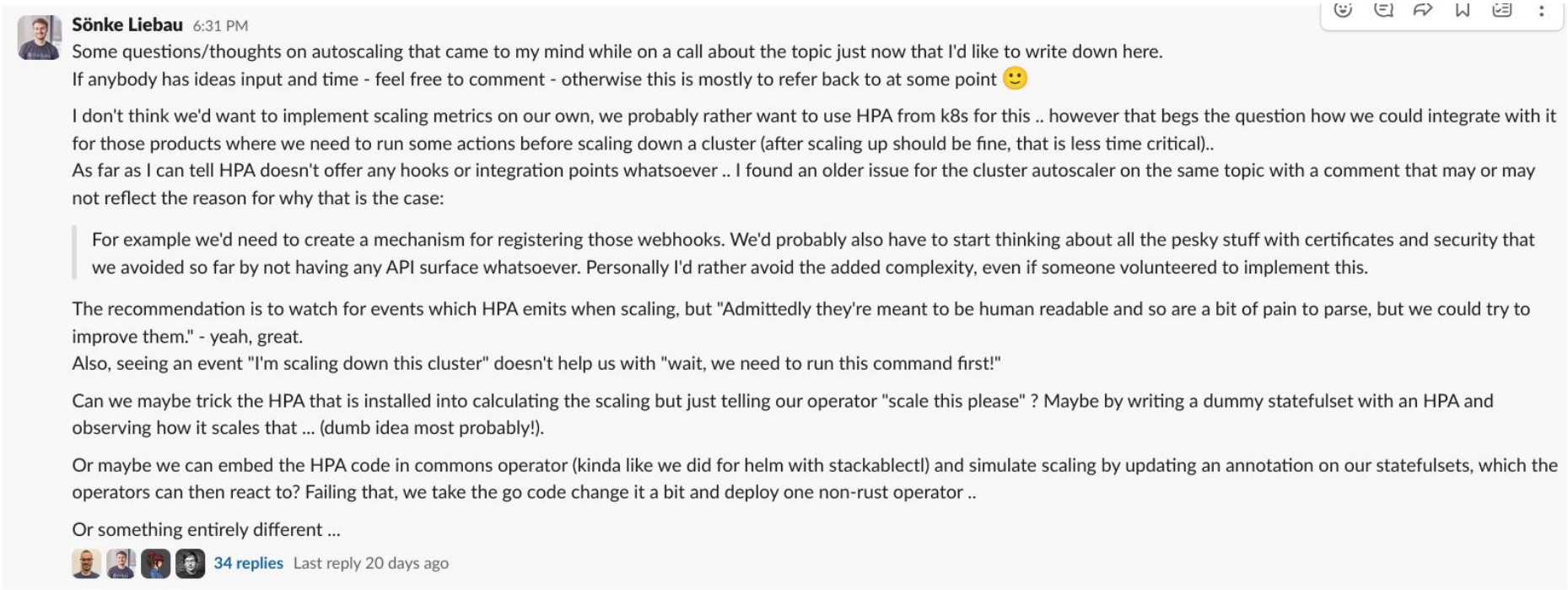

AUTOSCALING
AUTOSCALING EVERYWHERE
makeameme.org

Why am I Talking Right Now?



Sönke Liebau 6:31 PM

Some questions/thoughts on autoscaling that came to my mind while on a call about the topic just now that I'd like to write down here.
If anybody has ideas input and time - feel free to comment - otherwise this is mostly to refer back to at some point 🙂

I don't think we'd want to implement scaling metrics on our own, we probably rather want to use HPA from k8s for this .. however that begs the question how we could integrate with it for those products where we need to run some actions before scaling down a cluster (after scaling up should be fine, that is less time critical)..
As far as I can tell HPA doesn't offer any hooks or integration points whatsoever .. I found an older issue for the cluster autoscaler on the same topic with a comment that may or may not reflect the reason for why that is the case:

> For example we'd need to create a mechanism for registering those webhooks. We'd probably also have to start thinking about all the pesky stuff with certificates and security that we avoided so far by not having any API surface whatsoever. Personally I'd rather avoid the added complexity, even if someone volunteered to implement this.

The recommendation is to watch for events which HPA emits when scaling, but "Admittedly they're meant to be human readable and so are a bit of pain to parse, but we could try to improve them." - yeah, great.
Also, seeing an event "I'm scaling down this cluster" doesn't help us with "wait, we need to run this command first!"

Can we maybe trick the HPA that is installed into calculating the scaling but just telling our operator "scale this please" ? Maybe by writing a dummy statefulset with an HPA and observing how it scales that ... (dumb idea most probably!).

Or maybe we can embed the HPA code in commons operator (kinda like we did for helm with stackablectl) and simulate scaling by updating an annotation on our statefulsets, which the operators can then react to? Failing that, we take the go code change it a bit and deploy one non-rust operator ..

Or something entirely different ...

34 replies Last reply 20 days ago

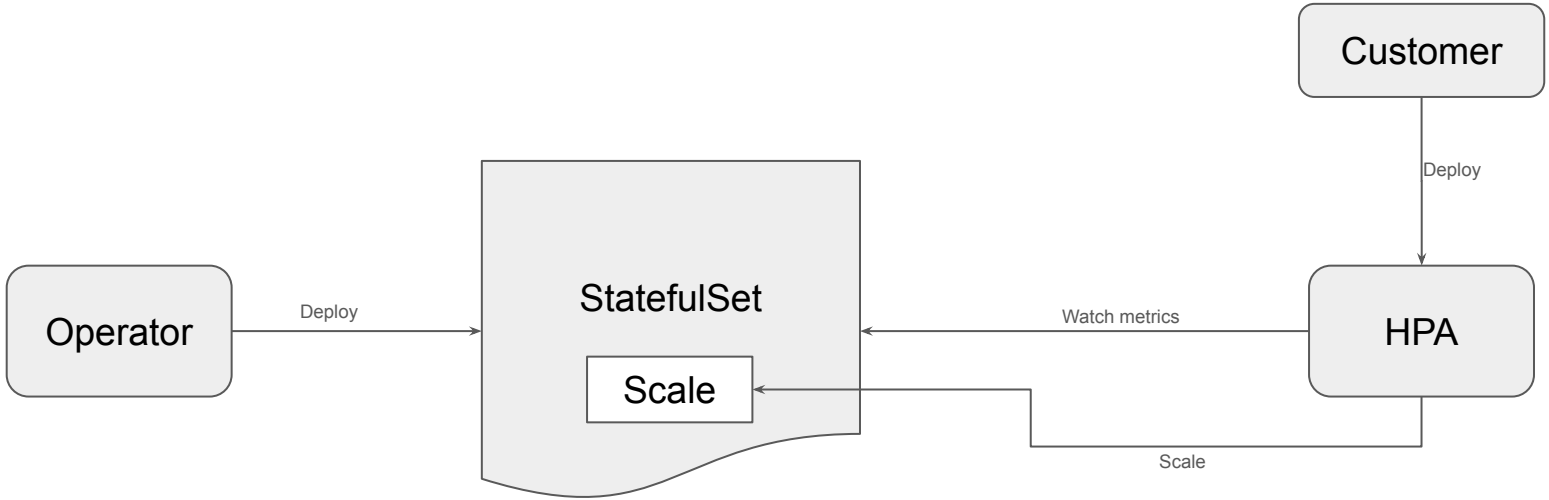
Current Situation

2. **Autoscaling**: Although not supported by the platform natively yet, you can use HorizontalPodAutoscaler to autoscale the number of Pods running for a given rolegroup dynamically based upon resource usage. To achieve this you need to omit the number of replicas on the rolegroup to be scaled, which in turn results in the created StatefulSet not having any replicas set as well. Afterwards you can deploy a HorizontalPodAutoscaler as usual. Please note that not all product-operators have implemented graceful shutdown, so the product might be disturbed during scale down. Later platform versions will support autoscaling natively with sensible defaults and will deploy HorizontalPodAutoscaler objects for you.

```yaml
---
apiVersion: trino.stackable.tech/v1alpha1
kind: TrinoCluster
metadata:
  name: simple-trino
spec:
  image:
    productVersion: "451"
  clusterConfig:
    catalogLabelSelector:
      matchLabels:
        trino: simple-trino
    listenerClass: external-unstable
  coordinators:
    roleGroups:
      default:
        ~~replicas: 1~~
  workers:
    roleGroups:
      default:
        ~~replicas: 1~~
```

- Omit replica count from CRD
- Deploy your own HorizontalPodAutoscaler to target StatefulSets deployed by operator

How it Works


Customer
Deploy
Operator
Deploy
StatefulSet
Scale
Watch metrics
HPA
Scale

Benefits

Pro:

- Easy for us, no implementation effort
- Flexible, just reuses existing K8s scaling functionality
- People are used to it (existing functionality)

Con:

- No hooks to run any potentially needed actions before or after scaling
- Easy to misuse, if replicas is accidentally filled the HPA fights with the operator

Requirements

Must:

- Allow us to control the scaling / run code before scaling happens
  - delay scaling until we are done!

Should:

- Use existing scaling mechanisms
- Disallow ambiguous configuration (op and scaler fighting with each other)

Could:

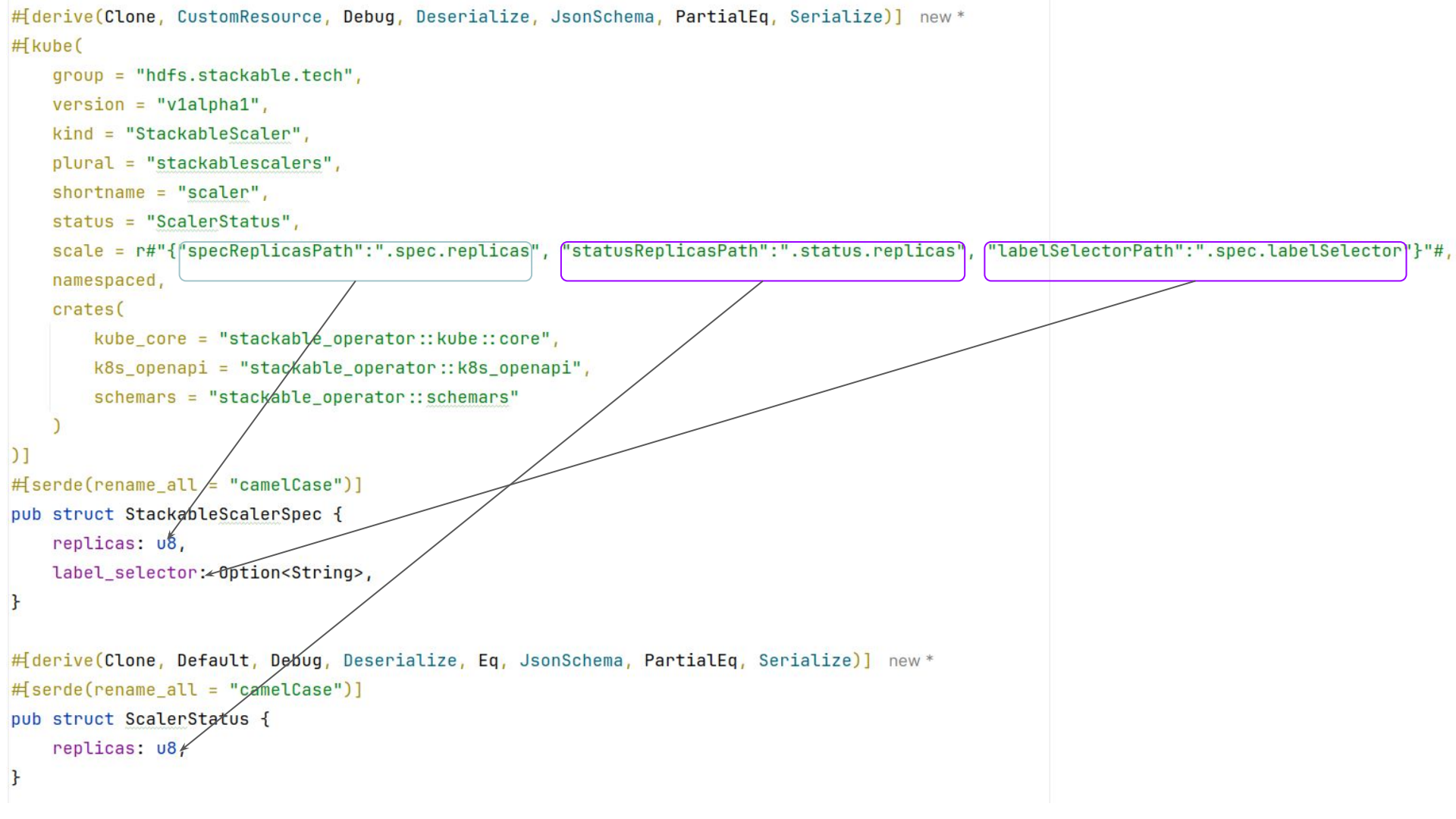
- Isolate as much as possible into operator-rs and have operators _just_ define necessary scaling steps

Proposal

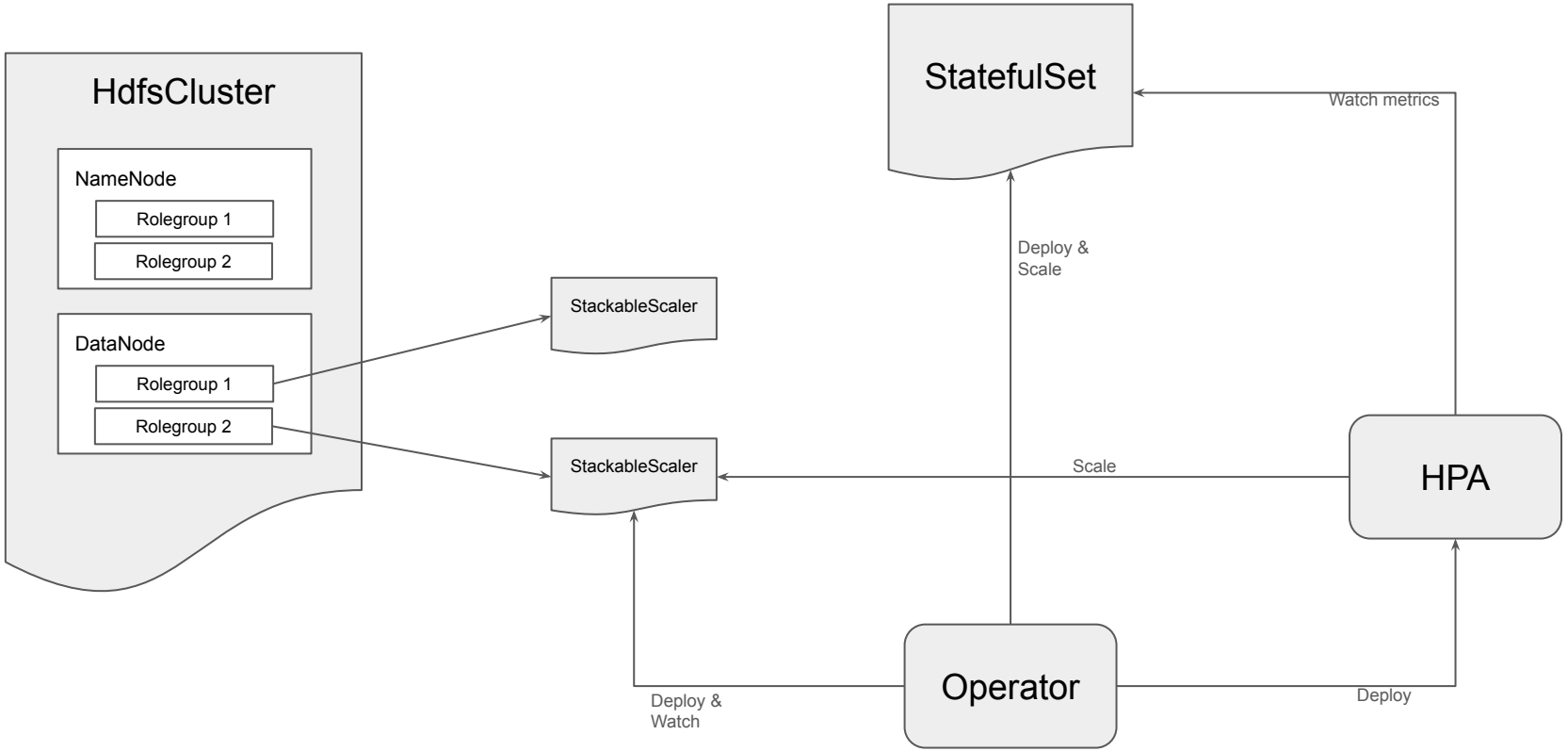
- Custom Resources can also be targets for the HorizontalPodAutoscaler, if they expose a ‘Scale’ subresource
- We could add a scale subresource to our ‘*Cluster’ crds and enable targeting these with HPAs
  - This would only allow scaling one rolegroup - for most products we can probably decide which role should be the scalable one (e.g. hdfs: datanodes, trino: workers, …)
  - Rolegroups cannot be distinguished here
- Introduce a separate resource that can be deployed for every roleGroup that should be scalable
  - Meet the StackableScaler

Possible CRD Changes (not a suggestion, just and idea!!)

```yaml
nameNodes:
  config:
    listenerClass: external-stable
  roleGroups:
    default:
      replicas:
        horizontalPodAutoscaler:
          # Vanilla K8s Object
```


#[derive(Clone, CustomResource, Debug, Deserialize, JsonSchema, PartialEq, Serialize)] new *
#[kube(
group = "hdfs.stackable.tech",
version = "v1alpha1",
kind = "StackableScaler",
plural = "stackablescalers",
shortname = "scaler",
status = "ScalerStatus",
scale = r#"{"specReplicasPath":".spec.replicas", "statusReplicasPath":".status.replicas", "labelSelectorPath":".spec.labelSelector"}"#,
namespaced,
crates(
kube_core = "stackable_operator::kube::core",
k8s_openapi = "stackable_operator::k8s_openapi",
schemars = "stackable_operator::schemars"
)
)]
#[serde(rename_all = "camelCase")]
pub struct StackableScalerSpec {
replicas: u8,
label_selector: Option<String>,
}
#[derive(Clone, Default, Debug, Deserialize, Eq, JsonSchema, PartialEq, Serialize)] new *
#[serde(rename_all = "camelCase")]
pub struct ScalerStatus {
replicas: u8,
}

How it Works


HdfsCluster
NameNode
Rolegroup 1
Rolegroup 2
DataNode
Rolegroup 1
Rolegroup 2
StackableScaler
StackableScaler
StatefulSet
Watch metrics
Deploy &
Scale
Scale
HPA
Operator
Deploy &
Watch
Deploy