Capacity Management For Fun & Profit

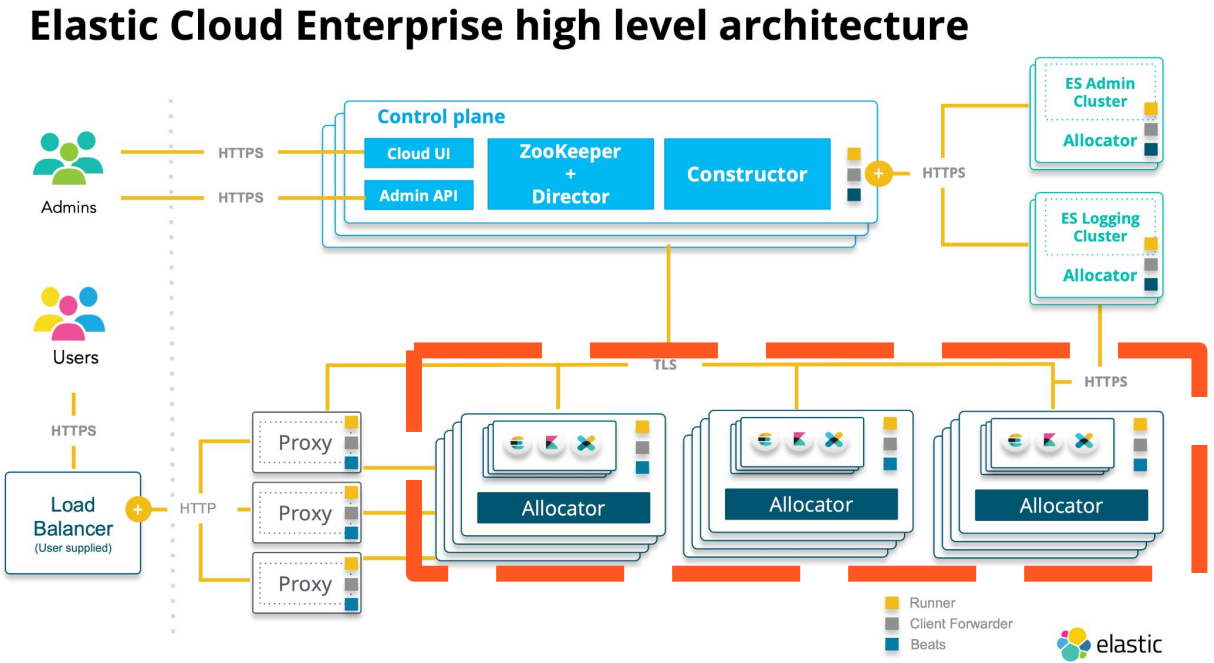
Aly Fulton (@sinthetix), Sr. Site Reliability Engineer @ Elastic

How it started

Well, I’m interested in capacity...

Hello, Elastic Cloud! 👋

- Elastic Cloud is a managed Elasticsearch service.
- Allocators contain the nodes for ES clusters.
- Nodes are scaled and sold in RAM increments.
- Each allocator is a VM.
- I make sure we have enough VMs and we use them efficiently.



ECE is the standalone product offering which evolved from the Elastic-hosted Cloud SaaS.

It's Cloudy ☁️

- Support deployments on AWS, GCP, Azure.
- Sept 2019, 16 regions available.
- Support up to 3 availability zones for high availability deployments.
- Used 4 VM archetypes:
    - High I/O, high storage, high memory, high CPU.

The E in SRE 👩‍💻

- We have internal tooling that handles some important capacity functions.
- Ran as manual or scheduled jobs.
- Written in Go.
- Maintained by SRE Tooling team.

The “soft” side 🧸

- Capacity needed to be greenfielded.
- Fell under Cloud SRE Infrastructure team.
  - Lone wolf of the team. 🦝
- Work with finance, analytics, product managers, and other SRE teams.
- Also work closely with CSPs.

Planning? Let's try managing, first.

We had no just-in-time capacity…

Autoscaling*

*Maybe we should just say scaling

State of mind 🧠

Stateful VMs also mean we can’t just decrease VM scale sets. Instead, we have to:

1. find the allocators we are no longer using.
2. find the associated VM and its scale set.
3. decrease scale set upon termination.

Up on the downside ↕️

- Old default cluster resizing method was “grow and shrink” so we needed free capacity for resizing cluster nodes (on top of their existing nodes).
- We also needed capacity for new clusters.
- We scale based on customer allocation to allocators.
    - Not a metric we can utilize built in CSP resource scaling policies with.

autoscaling.yml x3

- Autoscale configuration was per CSP, per region, per configuration
- Based on contiguous free RAM dubbed “blocks.”
- Increased when minimum block size not met.
- Decreased when maximum block size was exceeded.
- Job ran every 15-30 minutes.
- Monitored in a Kibana dashboard.

```
config:
  -
    min_block_unit: 1
    max_block_unit: 2
    block_unit_size: 58
    instance_configuration_id: aws.data.highio.i3
    region: aws-eu-central-1
```

Limitations in “auto” 

- Each autoscaling yaml file was hand updated.
- The numbers were best guess, not data-backed.
- Required a PR every time we needed to make a change:
  - CSP out of capacity? Make a change.
  - Black Friday? Make a change.
  - Copypasta error? Make a change.
- Tooling designed to pick a random allocator of the config type.
  - This sometimes meant getting the wrong VM size if we were switching sizes. (i3.8xl vs i3.2xl)

Autoscaling gone rogue! 🤠

- Autoscaling would sometimes infinitely spin if there was an incompatibility with block sizing.
- New allocators would not spin up properly and autoscaling would keep trying.
- A few times we only noticed because we got spammed with alerts from the CSP that they were out of capacity. (Sorry!)
- Info messages on spin ups but no warning alerts if block size mismatched or allocators failed to spin up properly.
- Other times tooling would not run and we'd only find out if there was an OOC alert.

Out of capacity ⚠️

- If customers requested more nodes than we had free blocks available, on call SRE would get paged to **manually** spin up capacity.
- It was a fun fight with our tooling which would autoscale down empty allocators before you could perform the plan change.
    - Yes, we eventually put “temporarily turn off the autoscaling” in the Runbook.
- It cannot be emphasized enough that this ***sucked*** for our SREs.

The other Covid surge ⚡

- Capacity demands increased.
- Out of capacity was frustrating for customers & our SREs.
- Worked with data analyst for smarter numbers.
- Updated the autoscaling yaml files to have enough capacity for our largest customers to double their clusters if necessary.
- This was a costly “just-in-case!”

Binpacking*

*Fragile, handle with care

Reduce, reuse ♻️

- Nodes often get moved off of allocators, creating empty space.
  - Trials expire.
  - Customers change their deployments.
- Want to use only as many allocators as necessary.
- Binpacking is the tooling that packs nodes in empty allocator space.
- Manually triggered jobs.
- The code is legacy af.

Please don't litter the allocators

Yay:

Zone	Allocated RAM capacity
us-east-2a	58 GB available of 58 GB
us-east-2a	9 GB available of 58 GB
us-east-2a	4 GB available of 58 GB
us-east-2a	3 GB available of 58 GB
us-east-2a	1 GB available of 58 GB
us-east-2a	1 GB available of 58 GB
us-east-2a	0 MB available of 58 GB
us-east-2a	0 MB available of 58 GB
us-east-2a	0 MB available of 58 GB
us-east-2a	0 MB available of 58 GB
us-east-2a	0 MB available of 58 GB
us-east-2a	0 MB available of 58 GB
us-east-2a	0 MB available of 58 GB
us-east-2a	0 MB available of 58 GB

Nay:

Zone	Allocated RAM capacity
us-east-2a	58 GB available of 60 GB
us-east-2a	58 GB available of 60 GB
us-east-2a	58 GB available of 60 GB
us-east-2a	58 GB available of 60 GB
us-east-2a	58 GB available of 60 GB
us-east-2a	44 GB available of 60 GB
us-east-2a	2 GB available of 60 GB

To the limit 🛑

- Algorithm will skip allocators with broken clusters or with plan changes in process.
- All of a cluster's individual nodes show as "plan change in progress" when any single node is being moved.
- This means you can only do one config and one AZ at a time, per region.
    - Manually!

We needed more “waste management.”

Tags*

*Helpers that hurt sometimes

Blocked & Unreported 🚫

- Turns out a tag so your tooling ignores allocators is a double-edged sword.
  - Would add tag so tooling wouldn't tear down large capacity requests fulfillments.
  - Ignored allocators do not get torn down when empty.
- Same with maintenance mode.
  - Allocators in maintenance mode do not accept new nodes.
  - They do not get torn down unless they are empty.

Ghosts*

*Hidden capacity costing us money

More going rogue ☠️

- Allocators would fail to bootstrap properly.
  - VMs would be spun up but not become allocators, remaining undetected by system.
  - Autoscaling would try to spin up allocators until it achieved minimum free block space.
  - Only indicator is comparing allocator and VM counts in Admin vs CSP.
  - Usually happened on staging, which is less monitored (but costs the same money!).

Toil*

*Every SRE’s favorite word

Manual operations 

- Had to manually spin up allocators to move outdated allocators' nodes to.
- Would often involve allocators sitting in maintenance for a while to make sure we had the capacity to move the old allocators' nodes to.
- Allocators vacated one by one by an SRE.
- Expensive on both infrastructure and people side.

Sometimes things are out of your control.

The Cloud*

*Does not just materialize when needed

Partly Cloudy 🌤

- AWS, GCP, and Azure are different.
  - Different APIs/tools.
  - Different processes.
  - Different instance types.
- Inside the Clouds are messy too:
  - Some regions only have two availability zones. Some have more than three.
  - Some regions do not support all VM types.
  - Some regions only support VM types in some availability zones.

Storm Clouds ⛈️

- Hard out of capacity.
  - We expand quickly, sometimes hitting CSP limits.
- GCP has quotas for everything.
  - Even on CUDs!
  - Great safety nets but annoying if unmonitored.
  - Can’t spin up allocators until additional quota added.
  - Stressful getting an OOC alert and having to wait until quota increase.
- No, we cannot just turn the cloud off for a second.
  - In Azure, if your VMSS cluster runs out of resources, only way to get more is to turn off all VMs to change clusters.

Capacity management is cost management!

How it’s going

46

Regions as of September 2021

Now for a tooling check-in!

I've never been good with silos.

Autoscaling*

*Now with more auto

Now with more to scale 📊

- 46 regions with new allocator types (and support for the old ones, too.)
  - Too big to memorize now.
- Still hard-coded in yaml
  - ...but at least adding them has been automated.
  - ...but you can still input the wrong information in the template.

Just in time! ⌚

- Our tooling now intercepts out of capacity messages and increases capacity on customer demand.
- We need less spare capacity on hand.
    - Which means less waste!
- We still have to have one of each type spun up in all three availability zones, in all regions, for our tooling to scale properly.
    - Still a lot of waste!

Infrastructure API 🤖

- The near future plan is to abstract our infrastructure into an API consumable by our tooling to be able to discover regions, new configurations, and variances in AZs automatically.
- This would allow our tooling to spin up new allocator types only when there is demand.
- Less room for human error in autoscaling, as infrastructure itself would be source of truth.

Binpacking*

*Not much has changed

Improvement needed 

- Algorithm could be made more efficient.
  - Prioritization of nodes that will be easier to move.
  - Recheck allocators that were skipped because of plan changes.
- Introduction of new types create more of a need for binpacking as people leave old types.
- The reward is high but so is the effort.

Tags*

*Now with more tags

Tag, you're it! ✋

- New tools to add tags that our tooling will respond to accordingly.
- This can help with removing too much accidentally spun up capacity.
- Helps with holding a bunch of normally constrained capacity for large customers.
- Also helps remove ignore or maintenance tags in bulk.

Don’t forget 🔔

- New alerting on allocators in maintenance for too long.
- Could still use alerting on ignored allocators.

Ghosts*

*No longer surprised

Ghost bustin' 🙈

- We now have a detection script!
  - Checks our inventory of allocators with VM inventory and isolates mismatch.
  - Identifies and gives option to terminate.
  - Separate tool, manually run locally.
- This will soon be obsolete because we will have alerting when allocators fail to be bootstrapped properly.
  - Prevention is always better!

Toil*

*All the way down

Fleet replacement 

- Now have rudimentary fleet replacement automation!
- Automates the one-at-a-time allocator's nodes move that was taxing to engineers.
- Still is inefficient and could use a better algorithm to decrease time to completion.
    - Similar to binpacking, better prioritization.
    - Improve concurrency.

The Cloud*

*Still does not just materialize

ARIMA to the rescue 

- We do real capacity planning now!
- Work even more closely with our CSP so they’re better aware of our growth plans.
    - Hardware does not grow on trees!
- We did have an issue where one default region changed and our projections were completely off for the swapped regions.
    - Context around data is important!
    - Communication is important!

Limiting limitations 🚷

- Still haven’t solved Azure VMSS limit problems.
  - We have to use multiple VMSS for the same allocator type.
  - Messy with how our tooling utilizes scale sets.
- Hoping new API can help us utilize multiple scale sets efficiently.

Catching quotas ☔

- Increasing GCP quotas is still manual.
  - This is a GCP thing.
- We do alert on quota consumption now.
- Really helpful to prevent OOC.

Lessons learned

Alerting, Monitoring

- Alert on capacity waste to save money on idle infrastructure.
  - Can automate remediation too.
- Alerting on quotas approaching limits largely prevents reaching said limit.
- Staging still costs money and deserves to be monitored and alerted on for large capacity offenses.
- Prevention > reaction.
- Simple alerts solve big problems.

Automation, Engineering

- Fundamental operations such as VM replacement and scaling should be automated.
- If you find yourself repeating tasks, even if it’s a special way, automate as much as possible.
- Eliminate space for human error.
- Engineering time is a resource with its own capacity limitations! Use it wisely!

Data-backed Decisions

- Guesses might serve you well in the beginning but real data serves you better.
- Easier to get leadership to agree with you with data to back your proposal.
- You still have to understand the nuances and context of your numbers to detect problems your alerting/monitoring might have missed.

Flexibility, Adaptability

- Rigid software does not adapt well to scale.
- Fix today’s problems in ways extensible to future problems.

Team Work, Communication

- Capacity work requires an understanding of customer and system patterns.
- There is a lot of advocacy work to discourage waste and encourage best practices across multiple teams.
- Silos hinder progress.
- Working cooperatively benefits everyone in the long run.

Capacity management is cost management!