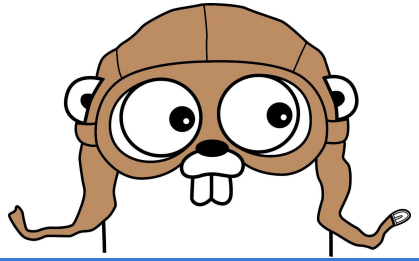


How we accidentally created a Cloud on our Cloud

Mofizur Rahman
Developer Advocate, IBM

Do Container Stuff,
Collect Stickers,
Write go code

@moficodes

What is Cloud

- Someone else’s computer
- On Demand
- Has way for users to access the service

The Problem

The Problem

- Run average of 10-20 workshop a month. (2-4 a week)
- Each workshop require avg of 30 K8s or OpenShift Cluster
- Spinning up and spinning down cluster resources is manual
- Cluster per subnet is limited
- There is a soft limit on volume per datacenter
- No clear way to collect workshop metrics
- The team that owns this process has 2 person

The Solution

Many ways to skin this cat

Worst Solution

Worst Solution

- Spin up clusters in the UI
- Manually give access to users in console

- Do That for every user for every workshop for every cluster
- Estimated time for the workshop lead + owner of the account 4 hour per workshop

Impact

- On average 2 work days spent for the account owner
- Pretty much at the upper limit on how many workshop we can support
- Huge cost center because resources needs to be created earlier and deleted later (Weekends, time zones)
- No higher level workshop

Passable (Not Bad) Solution

Passable Solution

- Use the CLI in a bash script.
- Update a config file for each workshop request
- Use the same config file to delete the resources
- Use a github repo for tracking cluster requests
- Simple webapp to handle workshop user access

Impact

- Significant improvement on previous solution
- The cluster access is now automated with the simple webapp
- Still a manual process. But since its a script takes less active time from dev advocates
- The actual workshop can scale to a fairly large number of attendees since cluster allocation is automated

Current Solution

Current Solution

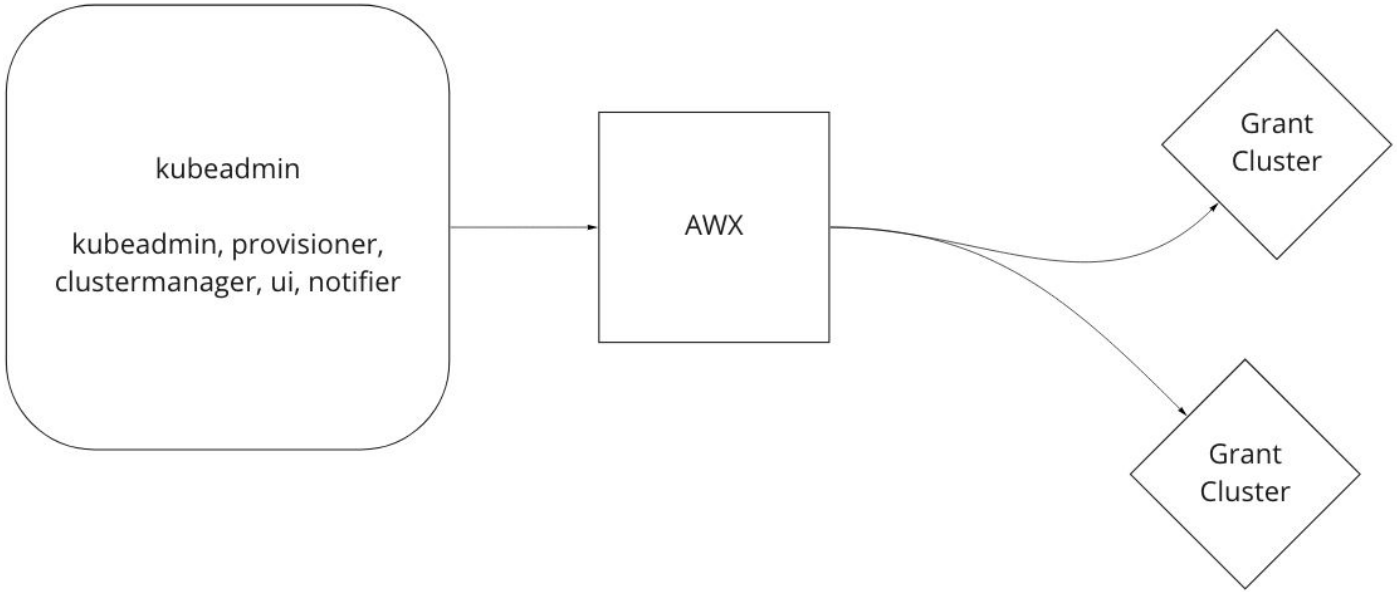
- UI to select options (React)
- API to talk to infrastructure (All Go)
- Some way (AWX) to spin up resources
- Some way to clean up resources
- Run post provisioning task on each cluster

What’s Lacking

- The UI is still a manual process. So weekend and time zones are still a problem.
- No real metrics collection from the workshop itself.
- No way to schedule creation of deletion of resources.

Impact

- UI makes it easy to teach anyone to spin up resources
- We can add retry logic or rate limit to request


kubeadmin
kubeadmin, provisioner, clustermanager, ui, notifier
AWX
Grant Cluster
Grant Cluster

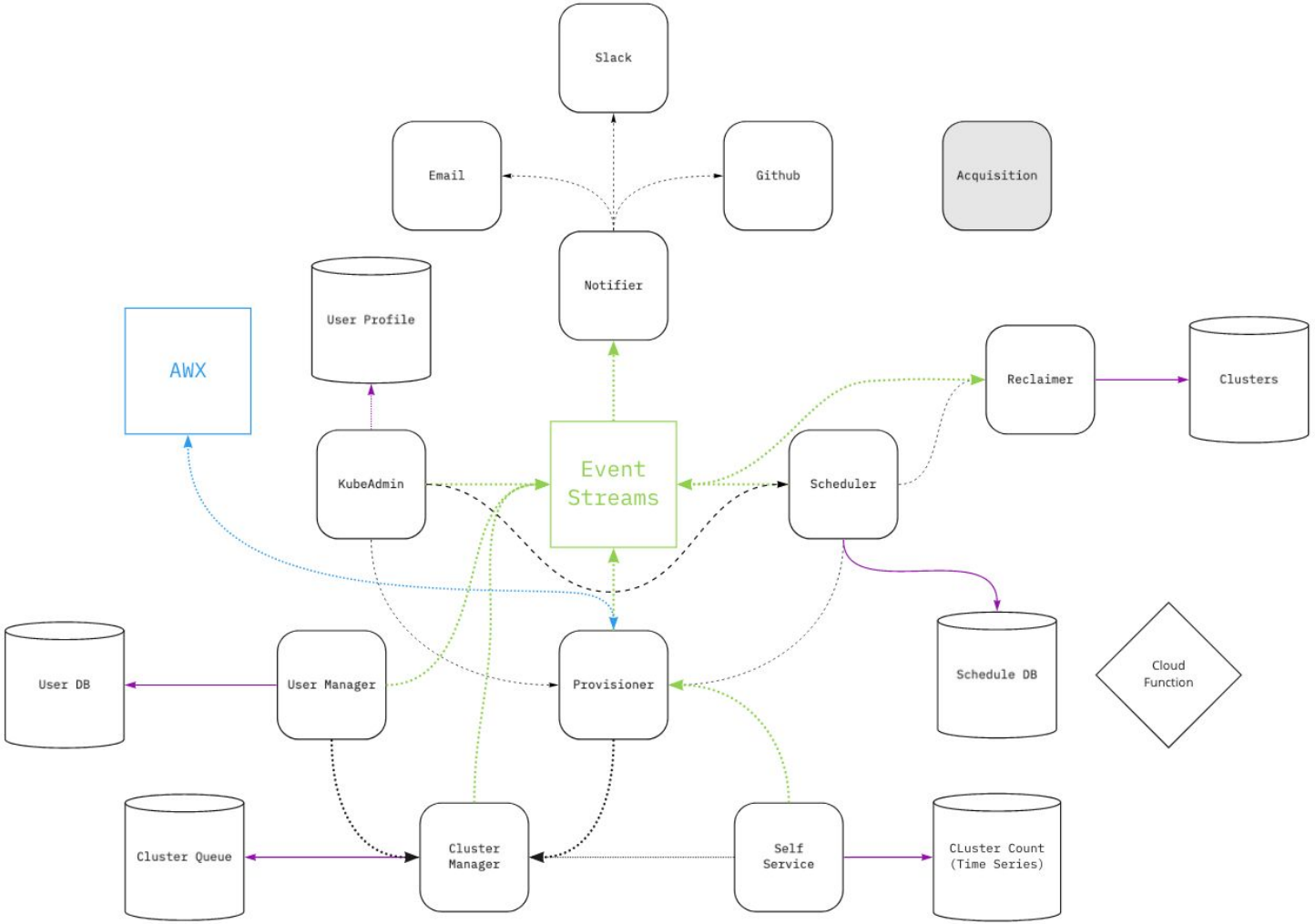
Ideal Solution

Ideal Solution

- Automated workshop request from github to cluster creation with approval in place
- Way to schedule creation and deletion of resources
- Open it up for other teams in org to run their resource creation
- Have proper metrics for workshop run on these accounts

Impact

- Anyone within the org would be able to use this. No more dependency on our small team.
- We would not need to spend cycle managing resources
- Cost saving with not running resources beyond what's needed
- Take better decisions when new workshops and events are considered.


Slack
Email
Github
Acquisition
Notifier
User Profile
AWX
Reclaimer
Clusters
KubeAdmin
Event Streams
Scheduler
User DB
User Manager
Provisioner
Schedule DB
Cloud Function
Cluster Queue
Cluster Manager
Self Service
CLuster Count (Time Series)

So why did we make a cloud?

- The features we needed are not needed by most people. It would not make sense to implement that in the public cloud interface.
- A cloud interface would be easier for us to use and scale it even for an internal audience

Should you build your own cloud interface?

- Does the cloud you have does not have a easy way to do what you need?
- Do you often find yourself writing custom code to do things in your cloud?
- Does other teams do the same things?
- Do you struggle to keep the resources in check?

If the answer to these questions are yes...

Then maybe you need to build a interface to your cloud.

But Infrastructure as Code First.

Infrastructure as Code

- Cloud Automation (Terraform, Pulumi, Ansible, Chef, Puppet)
- Deployment Automation (Ansible, Chef, Puppet, Pulumi)
- CI/CD (Github Actions, Travis, Jenkins, Harness, Argo, Tekton)

Rolling out a custom solution should be towards the bottom of your list.

Why should you consider building a cloud?

- Sometime reinventing the wheel is the best way
- A small script across different teams and orgs and needs become a big dependency
- Most teams should not have to own cloud resources
- Interface of the cloud of your choice might not have all the answers that you need

Considerations

- You can always do more. Do as much as necessary
- Cloud has a lot of API and they can change be ready to update things.
- Solving general cloud problem is probably going to be more than you can take on. Solving your and your adjacent teams problems are enough.
- Don’t start with recreating cloud. Consider alternatives first.

Mofizur Rahman
Developer Advocate, IBM

Do Container Stuff,
Collect Stickers,
Write go code

@moficodes