Infrastructure Review

Timeline & LOE

Required dates to completion

- Cost Optimization 19 Jul 2019
  - **DEVOPS-245** - Getting issue details... STATUS
- Budget for DevOps TBD
- Decide on replacement or update to 2.0 25 Jul 2019
  - LOE to get to 01 Dec 2019
  - Do we need additional resources

Scope

This is an effort to reconsider past decisions related to infrastructure, technology, and tooling selection in an effort to spec out possible changes to improve the AutoGravity Platform. What follows is meant to serve as a place to document and drive discussions related to proposed changes. In cases where no change is proposed, some technologies and or tools may pass without comment/discussion. For example, if no proposal is made to move away from containerized deployments, alternatives to containerization will not be discussed.

In each case of a proposed change, an effort should be made to justify the need/desire for the change and a basic argument should put forward to support the change.

Container Orchestration

Justification

Currently, Rancher 1.6 is used in two distinct ways. Firstly, the Rancher UI is used to enable AutoGravity engineers to view and in some cases change app configuration, i.e. update environment variables, change logging levels, etc, without the need to an AWS account, or direct access to docker hosts. Secondly, Rancher Cattle is used to orchestrate the lifecycle, networking, monitoring, and redundancy of containers.

Rancher 1.6 has been deprecated in favor of Rancher 2.0 and only receives extremely critical security patches. Rancher 1.6 maintains only bare-bones support as an incentive to convince users to upgrade to 2.0. This effectively means that Rancher 1.6 and the Cattle orchestrator are technological dead ends with only a short term future. Because of this, it is advantageous for Rancher 1.6/Cattle users to plan an upgrade or migration to an alternative sooner rather than later.

Links

- https://rancher.com/docs/rancher/v2.x/en/faq/#how-does-rancher-v2-x-affect-cattle
- https://rancher.com/support-maintenance-terms/
- https://github.com/rancher/rancher/issues/16719#issuecomment-457523306
- https://rancher.com/docs/rancher/v1.6/en/
- https://rancher.com/docs/rancher/v2.x/en/

Selection Criteria

The selected replacement of Rancher 1.6 and Cattle should be expected to meet the following requirements.

Though the candidate itself may not meet every requirement via a core feature, there should be a way to meet each requirement by integration with a different tool/framework/technology.

1. Deployment of containers from a private image registry
2. Controlling the redundancy and availability of containers over their lifetime (i.e. not just at deploy time)
3. Restarting containers should it become unhealthy or unresponsive (based a defined configuration)
4. Scaling containers in/out (based on defined configuration)
5. Movement/rescheduling of containers should a container host become unhealthy or unresponsive
6. Management of containers network access and the visibility of the container to the outside world
7. Load balancing between container instances
8. Well documented API

In addition to the criteria above a candidate may be able to offer the following as a bonus, but not necessarily required feature

- Intuitive UI to display and modify container configuration

Non-Feature Selection Criteria

The selected replacement of Rancher 1.6 and Cattle should meet the following criteria:

1. The selected candidate should be reasonably easy to deploy and manage.
2. the selected replacement should be deployable to AWS unless it is a completely managed service, in which case it should integrate well with AWS offerings.
3. The selected replacement should have a reasonably active community of all support is provided by an open-source community.
4. The selected replacement should have quality support plans (at an acceptable price - price range may need clarification before selection) if support is provided by a third party company.

Alternatives Considered

AWS Fargate

Overview

Per Amazon:

AWS Fargate is a compute engine for Amazon ECS that allows you to run containers without having to manage servers or clusters. With AWS Fargate, you no longer have to provision, configure, and scale clusters of virtual machines to run containers. This removes the need to choose server types, decide when to scale your clusters or optimize cluster packing. AWS Fargate removes the need for you to interact with or think about servers or clusters. Fargate lets you focus on designing and building your applications instead of managing the infrastructure that runs them.

Pros

- No nodes (servers) to manage

Cons

- Less customization
- Cost is directly proportional to CPU and memory usage, which can increase the cost for poorly optimized workloads
- Support with existing continuous deployment tools is relatively new.

Amazon EKS

Amazon Elastic Container Service for Kubernetes (Amazon EKS) makes it easy to deploy, manage, and scale containerized applications using Kubernetes (k8s) on AWS. Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications. It is the industry standard in deploying containers to production.

Overview

Pros

- TBD

Cons

- AWS is regularly a few versions behind K8S releases

Amazon EKS + Rancher 2.0

Overview

Rancher is a complete software stack for teams adopting containers. It addresses the operational and security challenges of managing multiple Kubernetes clusters while providing DevOps teams with integrated tools for running containerized workloads.

Pros

- K8S has well-known resiliency/healing/deployment patterns baked in. We should be able to leverage health probes and deployment strategies with little effort
- K8S has a sizable community, learning resources
- We are not responsible for managing K8S master nodes

Cons

- AWS is regularly a few versions behind K8S releases
- Friction between K8S created resources (ingress load balancers, etc) and terraform state file, i.e can block clean terraform destroy operations if all K8S objects are not deleted first
- Rancher 2.0 UI seems to have taken a step down in quality from the 1.6 UI
- Rancher 2.0 UI seems less able to provide users without aws/cluster/kubectl access a decent experience
- Rancher 2.0 documentation seems less detailed than 1.6

Github Repository

https://github.com/AutoGravity/EKS-POC

Open Questions

1. Rancher 2.0 uses **etcd** as a backing store instead of a traditional database. We need to have a plan in place to treat the **etcd** store as a persistent volume outside of an individual host mount (EFS, ?, ?).
2. Rancher 2.0 is capable of creating EKS clusters from the UI. We need to determine internal best practices for cluster creation (i.e. Terraform and import into rancher vs Create from Rancher and import to Terraform.)
3. Determine best practice related to running Rancher outside of Kubernetes, or within the cluster, it is managing.
4. We should test both ASG scaling of workers as well as the K8S Horizontal Pod Autoscaler in order to define our scaling strategy
5. We need to determine how to best monitor K8S workers.

NetFlix Titus

Overview

Pros

- TBD

Cons

- TBD

Mesos DC/OS

Overview

Pros

- TBD

Cons

- TBD

TODO: Each of the above deserves a treatment of pro/con and listing of
concerns related to each candidate in order to further drive the discussion.
Additionally, any candidates left off of the list should be added.

Continuous Delivery

Justification

Currently, AG services are deployed using custom-built python scripts directly
integrating with Rancher 1.6 via the v2-beta api. This means that the deployment
automation is extremely tied to Rancher 1.6 and requires new development effort
for changes in deployment strategy, or feature additions, and bug fixes. This
need for active maintenance and new development slows down AutoGravity's ability
to deliver when changes are needed to the deployment automation as new service deployments may wait
until the deployment automation is updated and qualified.

The point above in confluence with the need to move away from Rancher 1.6 means
that we must either rework existing deployment automation in-house or Consider
adopting an existing tool which provides a mature and community validated
continuous delivery framework. Adoption of such a tool would relieve AutoGravity
engineers of the development responsibility currently needed to extend or fix
deployment automation, as well as providing the organization with ready access to
new features and deployment strategies targeting multiple clouds and orchestrators.

Alternatives Considered

spinnaker

Overview

Spinnaker is an open-source, multi-cloud continuous delivery platform for releasing software changes
with high velocity and confidence. Created at Netflix, it has been battle-tested in production by
hundreds of teams over millions of deployments. It combines a powerful and flexible pipeline
management system with integrations to the major cloud providers.

Pros

- Open Source - free!
- Native blue/green and canary deploy capabilities.
- Wide community support.

Cons

- Difficult to set up and maintain.

harness

Overview

Harness is the first Continuous Delivery-as-a-Service platform that uses Machine Learning to simplify the entire process of delivering code from artifact into production – quickly, safely, securely, and repeatably.

Pros

- Completely managed platform

Cons

- Expensive

armory

Overview

Armory Spinnaker is a pre-configured distribution of Spinnaker that runs in a Kubernetes cluster. Spinnaker can helps coordinate and orchestrate complicated tasks with pipelines for deploying software.

Why Pay for Armory When Spinnaker is Free?

Pros

- Completely managed platform

Cons

- Paid version of spinnaker which is free.

Other

The below are tools that we are a team are interested in using but does not fall under a specific category.

- **Ansible**
- **Terraform by HashiCorp**
- **Docker**

helm

Overview

Helm is a tool for managing Kubernetes charts. Charts are packages of pre-configured Kubernetes resources. It streamlines installing and managing Kubernetes applications. Think of it like apt/yum/homebrew for Kubernetes.

Pros

- TBD

Cons

- TBD

TODO: Each of the above deserves a treatment of pro/con and listing of concerns related to each candidate in order to further drive the discussion. Additionally, any candidates left off of the list should be added.

Budget Evaluation

Evaluate what tools to keep within a given budget

- **Third-Party Vendors, Packages, Tools and Services**

Related Tickets

- **DEVOPS-256** - Getting issue details... STATUS
- **DEVOPS-251** - Getting issue details... STATUS
- **DEVOPS-245** - Getting issue details... STATUS