Spinnaker-Concepts

29 January 2021 10:18

Problem:

Continuous delivery and continuous deployment rely on the ability to define an automated and repeatable process for releasing updates. If we have as high as tens of releases per week it quickly becomes untenable for each version to be manually deployed in an ad hoc manner. What teams need are tools that can reli- ably deploy releases, help with monitoring and management

Spinnaker is developed to address these issues.

- Spinnaker is an open-source, multi-cloud continuous delivery platform that helps you release software changes with high velocity and confidence

- **Spinnaker** is a free and open-source continuous delivery software platform originally developed by Netflix and extended by Google.
- It is designed to work with Kubernetes, Google Cloud Platform, AWS, Microsoft Azure and Oracle Cloud.

- Spinnaker was developed by Netflix as a successor to the internally developed Asgard. It was released under the Apache License 2.0 on November 16, 2015.

Spinnaker provides two core sets of features:

- **Application management**
- **Application deployment**

Application Management

- Application management features to view and manage your cloud resources
- Applications, clusters, and server groups are the key concepts Spinnaker uses to describe your services.
- Load balancers and firewalls describe how your services are exposed to users.

Application

- An application in Spinnaker is a collection of clusters, which in turn are collections of server groups. The application also includes firewalls and load balancers.
- An application represents the service which you are going to deploy using Spinnaker, all configuration for that service, and all the infrastructure on which it will run.

Cluster

- Clusters are logical groupings of Server Groups in Spinnaker.
- Note: cluster, here, does not map to a Kubernetes cluster. It's merely a collection of Server Groups, irrespective of any Kubernete s clusters that might be included in your underlying architecture.

Server Group

- The base resource, the Server Group, identifies the deployable artifact (VM image, Docker image) and basic configuration settings such as number of instances, autoscaling policies, metadata, etc.
- This resource is optionally associated with a Load Balancer and a Firewall.
- When deployed, a Server Group is a collection of instances of the running software (VM instances, Kubernetes pods).

Load Balancer

- A Load Balancer is associated with an ingress protocol and port range. It balances traffic among instances in its Server Groups.
- Optionally, you can enable health checks for a load balancer, with flexibility to define health criteria and specify the heal th check endpoint.

Firewall

- A Firewall defines network traffic access. It is effectively a set of firewall rules defined by an IP range (CIDR) along with a communic ation protocol (e.g., TCP) and port range.

Application Deployment

- Application deployment features to construct and manage continuous delivery workflows.

Pipeline

- Key deployment management construct in Spinnaker.
- Consists of a sequence of actions, known as stages.
- Can pass parameters from stage to stage along the pipeline.
- Can start a pipeline manually, or can configure it to be automatically triggered by an event, such as a Jenkins job completin g, a new Docker image appearing in your registry, a CRON schedule, or a stage in another pipeline.
- Pipeline notification by email, Slack,SMS for pipeline status

.

Stage

- Collection of sequential Tasks and composed Stages that describe a higher -level action the Pipeline will perform either linearly or in parallel.
- Can sequence stages in a Pipeline in any order, though some stage sequences may be more common than others.
- Spinnaker provides a number of stages such as Deploy, Resize, Disable, Manual Judgment, and many more.

Task

- A *Task* in Spinnaker is an automatic function to perform.

Deployment Strategies

- Spinnaker supports all the cloud native deployment strategies including Red/Black (a.k.a Blue/Green), Rolling red/black and C anary deployments, etc

Spinnaker Architecture

Spinnaker is composed of a number of independent microservices:

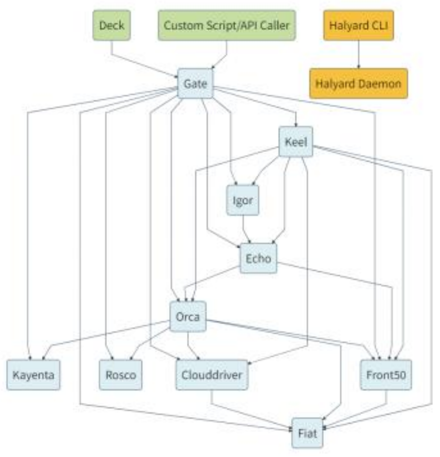
- **Deck** is the browser-based UI.

- **Gate** is the API gateway.
  The Spinnaker UI and all api callers communicate with Spinnaker via Gate.

- **Orca** is the orchestration engine. It handles all ad-hoc operations and pipelines. Read more on the Orca Service Overview.

- **Clouddriver** is responsible for all mutating calls to the cloud providers and for indexing/caching all deployed resources.

- **Front50** is used to persist the metadata of applications, pipelines, projects and notifications.
- **Rosco** is the bakery. It produces immutable VM images (or image templates) for various cloud providers.
  It is used to produce machine images (for example GCE images, AWS AMIs, Azure VM images). It currently wraps packer, but will be expanded to support additional mechanisms for producing images.
- **Igor** is used to trigger pipelines via continuous integration jobs in systems like Jenkins and Travis CI, and it allows Jenkins/Travis stages to be used in pipelines.
- **Echo** is Spinnaker's eventing bus.
  It supports sending notifications (e.g. Slack, email, SMS), and acts on incoming webhooks from services like Github.
- **Fiat** is Spinnaker's authorization service.
  It is used to query a user's access permissions for accounts, applications and service accounts.
- **Kayenta** provides automated canary analysis for Spinnaker.
- **Keel** powers Managed Delivery.
- **Halyard** is Spinnaker's configuration service.
  Halyard manages the lifecycle of each of the above services. It only interacts with these services during Spinnaker startup, updates, and rollbacks.



Spinnaker Installation and Configuration.

This section describes how to install and set up Spinnaker so that it can be configured for use in production.
We will install Spinnaker and set up CI/CD pipeline in following environments.

- Local Kubernetes cluster
- Google Kubernetes Engine.