Gaia Spark analysis platform

Technology choices

D Morris 1st Feb 2021

Original GDAF system

Zeppelin & Spark deployed on physical hardware

Based in part on Cloudera deployment

Initial goal for our project

- replicate GDAF in the cloud

Subsequent goals

- additional analysis tools
- scalable deployment
- more users
- more data
  - DR3
  - DR4

Gaia DPAC
Data Processing & Analysis Consortium

Gaia Data Analytics Framework (GDAF) description

D.Morris
Institute for Astronomy,
Edinburgh University

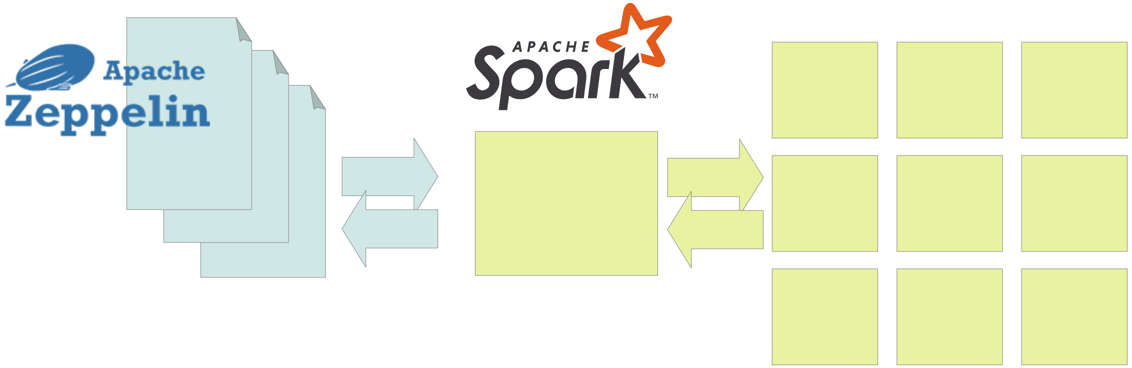
Technology choice #1
Spark job scheduler

Hadoop/Yarn

“known unknowns”

- GDAF system – working example
- Cloudera deployment - lots of documentation
- Standard deployment - lots of blogs and howtos

- Virtual machine based
- Spark cluster deployed on a static set of resources
- Zeppelin notebooks all interact with the same Spark cluster



Technology choice #1
Spark job scheduler

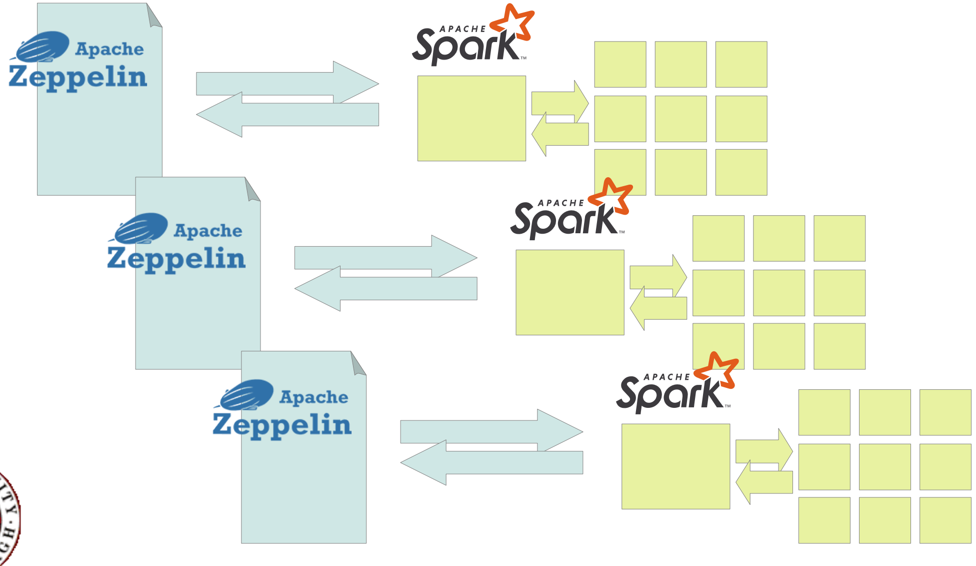
kubernetes

"unknown unknowns"

- Experimental in 2020
- Zeppelin pre-release only
- Not recommended for production

- Kubernetes container based
- Standard technology in 2021/2022

- Spark cluster on demand
- Notebooks launch their own Spark cluster



D.Morris
Institute for Astronomy,
Edinburgh University

Technology choice #1
Spark job scheduler

Developing both systems in parallel

Hadoop/Yarn

- Up and running early
- Spark and Zeppelin by Nov 2019
- Gaia DR2 parquet data by Dec 2019

- Automated with Ansible

Working system to develop science cases

- Python libraries
- AXS extension
- Parquet partitions

kubernetes

- Development platform for Spark and Zeppelin on Kubernetes

- Automated with Helm

Technology experiments

- Openstack Magnum & Manila
- Kubernetes Helm
- Terraform

- OAuth login
- Drupal CMS

D.Morris
Institute for Astronomy,
Edinburgh University

Technology choice #2
deployment tools

openstack

- REST webservice interface
- Good documentation
- Python command line client
- jq JSON parser

Working towards full automation

- always use command line
- always keep detailed notes
- always repeat the steps

Avoid the Horizon GUI as much as possible

Build a set of copy/paste scripts for each step

Automated with Openstack client before converting to Ansible

Create scripts for specific components

- Create script for Ceph router
- Create script for Ceph share
- Create script for SSH key pair

delete-all script to nuke everything from orbit

- Delete all Magnum clusters
- Delete all servers
- Delete all routers
- Delete all subnets
- Delete all networks

Technology choice #2 deployment tools

Working towards full automation

- always use command line
- always keep detailed notes
- always repeat the steps

Automated one step at a time.
Start with Openstack client before converting to Ansible.

Ansible playbooks for Hadoop/Yarn components

- Internal ssh keys
- Internal dns hosts
- Cinder volumes, btrfs filesystem, /etc/fstab device
- Manila shares, CephFS fuse mount
- Spark cluster - master and workers
- Zeppelin server

Good documentation

- Ansible website
- StackOverflow
- Tech blogs

Hadoop/Yarn create-all script

- 10% Openstack
- 90% Ansible

Combined playbook to create all the components

~30min to create everything from scratch

Technology choice #2
deployment tools

Working towards full automation

- always use command line
- always keep detailed notes
- always repeat the steps

Simple and easy cluster management

- My first cluster in < 5min
- Integrated with Openstack node autoscaler
- Integrated with Openstack network loadbalancer

Simple = few controls

- Homogeneous cluster, all the nodes are the same shape and size

Managed system

Managed = few controls

- When it works, everything is simple
- When it fails, you are not part of the loop

Technology choice #2
deployment tools

Working towards full automation

- always use command line
- always keep detailed notes
- always repeat the steps

Automated one step at a time.
Start with kubectl deployments
before converting to Helm.

Helm charts for all our Kubernetes components

- Ceph-CSI
- Manila-CSI
- Nginx ingress
- K8s dashboard
- Zeppelin
- Drupal

Writable charts directory

- Problems if charts are executed from local source
- Helm renames charts to chartmp during execution
- Breaks symlinks and volume mounts

Kubernetes create-all script

- 10% Openstack
- 90% Helm

Combined playbook to create all the components

- 10min to create everything from scratch

Technology choice #2
deployment tools

Terraform

Popular platform for cloud deployment

- In theory, abstract declarative form is portable
- In reality >50% of the code is platform specific

- Create router - needs uuid for target network and subnet
- Create server - needs uuid for flavor, size, network etc.

- Requires Openstack client calls to get the uuid values
- One last step to create the component with Openstack

Working towards full automation

- always use command line
- always keep detailed notes
- always repeat the steps

Tracks the deployed state in local files

- Fragile and easy to loose
- Users responsible for sharing
- Forget one step, hard to recover

Secrets are stored clear text in state files

- Sharing the state shares all the secrets, for the whole system
- Extremely easy to slip up and publish by accident

Technology choice #2
deployment tools

Create and delete scripts

Openstack delete-all

- Magnum delete can fail or leave parts behind
- Ansible delete can fail if ssh access is broken
- Ceph router needs to be deleted separately

delete-all nukes ***everything*** from orbit

Working towards full automation

- always use command line
- always keep detailed notes
- always repeat the steps

Hadoop-yarn create-all

- 90% automated
- 80% Ansible
- 10% Openstack - Ceph router, Ceph info
- 30min to create everything

Kubernetes create-all

- 90% automated
- 80% Helm
- 10% Openstack - Magnum cluster, Ceph router
- 10min to create everything

Three identical Openstack clouds

- gaia-dev - Kubernetes development
- gaia-test - Hadoop/Yarn integration
- gaia-prod - Live deployment for science

Technology choice #2 deployment tools

Working towards full automation

- always use command line
- always keep detailed notes
- always repeat the steps

openstack

- Good documentation
- Simple client interface
- Machine readable results

ANSIBLE

- Well documented
- Well supported on StackOverflow
- Lots of plugins

HELM | kubernetes

- De facto standard for Kubernetes deployments
- Good version control for dependencies

Terraform

- Didn’t live up to the hype
- Relies on local state
- Clear text secrets