Advanced Features & Emerging Technologies for Big Data

Big Data on Kubernetes - [Day 5]

LUAN MORENO

CEO & Data Architect
Data Engineer & MVP

MATEUS OLIVEIRA

Big Data Architect
Data In-Motion Specialist

Table of Content

Deploying a Production-Ready Kubernetes Cluster

Use of Best Practices and Resources to Deploy a Managed Kubernetes Cluster

Managed Kubernetes Cluster

- **Use of Cloud Native Approach**
- Build a Managed Cluster using IaC
- Split Resources by **Node Pools**
- Use of **Node Affinity** and **Taints**
- Consider Usage of Spot Instance

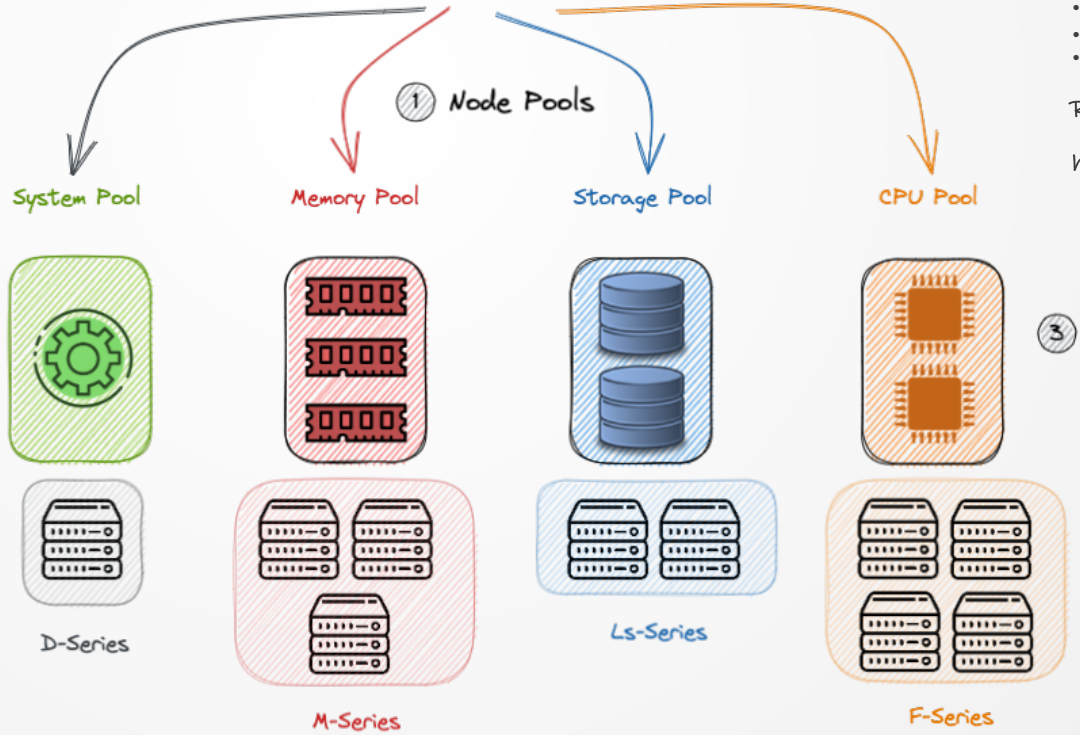
Managed Options = GKE, AKS & EKS

Self-Hosted Production Kubernetes Cluster

- **Use of Kubespray Tool**
- Bare Metal and Virtual Machines
- Avoid a Single Point of Failure
- Plan for Scalability
- Implement RBAC Rules
- Manage Load Balancer
- Span in Multiple Zones

Recommendation = Use for Lab Purposes

https://kubespray.io/



Spot Instances

- Node Pool Backed by a Spot Virtual Machine Scale Set
- Unutilized Capacity in Azure
- Significant Cost Savings
- **Without SLA**

Demo

Advanced Configurations for Big Data Products on Kubernetes

Apply Variety of Techniques to Enhance Deployment Process
of Big Data Applications

Labels & Selectors

- Key/Value Pairs Attached to Objects
- Specify Identifying Attributes
- Organize and Select Subsets of Objects

```
apiVersion: v1
kind: Pod
metadata:
  name: label-demo
  labels:
    environment: production
    app: nginx
spec:
  containers:
  - name: nginx
    image: nginx:1.14.2
    ports:
    - containerPort: 80
```

Resource Requests & Limits

- Specify Amount of Resources
- CPU - Units of CPUs
- Memory - Units of Bytes

```
apiVersion: v1
kind: Pod
metadata:
  name: frontend
spec:
  containers:
  - name: app
    image: images.my-company.example/app:v4
    resources:
      requests:
        memory: "64Mi"
        cpu: "250m"
      limits:
        memory: "128Mi"
        cpu: "500m"
```

Readiness & Liveness Probes

- Readiness = Know When a Container is Ready
- Liveness = Know When Restart a Container

```
readinessProbe:
    tcpSocket:
      port: 8080
    initialDelaySeconds: 5
    periodSeconds: 10
  livenessProbe:
    tcpSocket:
      port: 8080
    initialDelaySeconds: 15
    periodSeconds: 20
```

Pod Disruption Budgets

- Node Out-of-Resources
- Voluntary or Involuntary Disruptions
- Hardware, Deletion, Hypervisor, Kernel & Network
- PDB - Number of Replicas Application Tolerates

```
apiVersion: policy/v1
kind: PodDisruptionBudget
metadata:
  name: zk-pdb
spec:
  minAvailable: 2
  selector:
    matchLabels:
      app: zookeeper
```

Pod Presets

- Inject Information at Creation Time
- Secrets, Volumes, Mounts, Variables
- Use of Label Selector

```
apiVersion: settings.k8s.io/v1alpha1
kind: PodPreset
metadata:
  name: allow-database
spec:
  selector:
    matchLabels:
      role: frontend
  env:
    - name: DB_PORT
      value: "6379"
```

Affinity & Anti-Affinity

- Use of Node Selector using Labels
- Node Selector = Field of Pod Spec
- Affinity - Eligible for Scheduling
- Anti-Affinity - Constrain

```
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: kubernetes.io/e2e-az-name
            operator: In
            values:
            - e2e-az1
            - e2e-az2
```

Tolerations

- Taints and Tolerations
- Allow Pod Schedule onto Nodes with Matching Taints
- Ensure Pods are Not Scheduled onto Inappropriate Nodes

```
kubectl taint nodes node1 key1=value1:NoSchedule
```

```
tolerations:
- key: "key1"
  operator: "Equal"
  value: "value1"
  effect: "NoSchedule"
```

Demo

Scaling Big Data Applications on Kubernetes

Different Approaches to Scale-Up and Out Big Data Applications Efficiently on Kubernetes

VPA - Vertical Pod AutoScaler

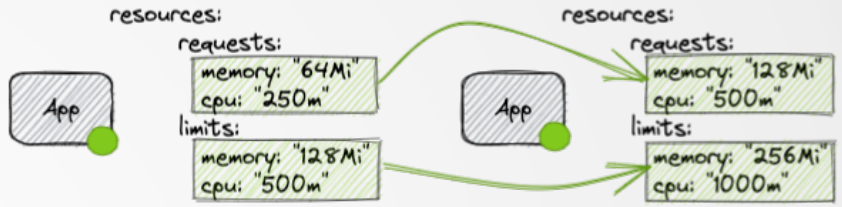
- Frees Users from Setting Up-to-Date Resource Limits and Requests
- Set Requests Automatically Based on Usage and Allow Proper Scheduling
- Maintain Ratios Between Limits and Requests Based on Utilization
- Configured with a Custom Resource Definition Object on Kubernetes

https://github.com/kubernetes/autoscaler/tree/master/vertical-pod-autoscaler

Manifest for VPA

```
apiVersion: autoscaling.k8s.io/v1
kind: VerticalPodAutoscaler
metadata:
  name: my-app-vpa
spec:
  targetRef:
    apiVersion: "apps/v1"
    kind:       Deployment
    name:       my-app
  updatePolicy:
    updateMode: "Auto"
```

Vertical Pod Autoscaler



HPA - Horizontal Pod AutoScaler

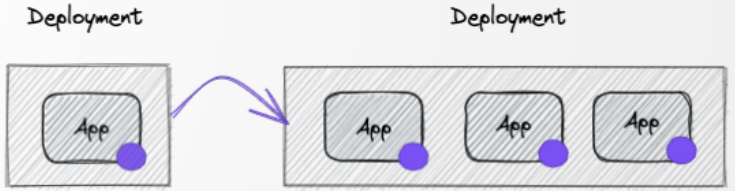
- Automatically Updates a Workload Resource - Deployment and Statefulset
- Scale Based on Demand and Usage = Deployment of New Pods

https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/

Manifest for HPA

```
behavior:
  scaleDown:
    stabilizationWindowSeconds: 300
    policies:
    - type: Percent
      value: 100
      periodSeconds: 15
  scaleUp:
    stabilizationWindowSeconds: 0
    policies:
    - type: Percent
      value: 100
      periodSeconds: 15
```

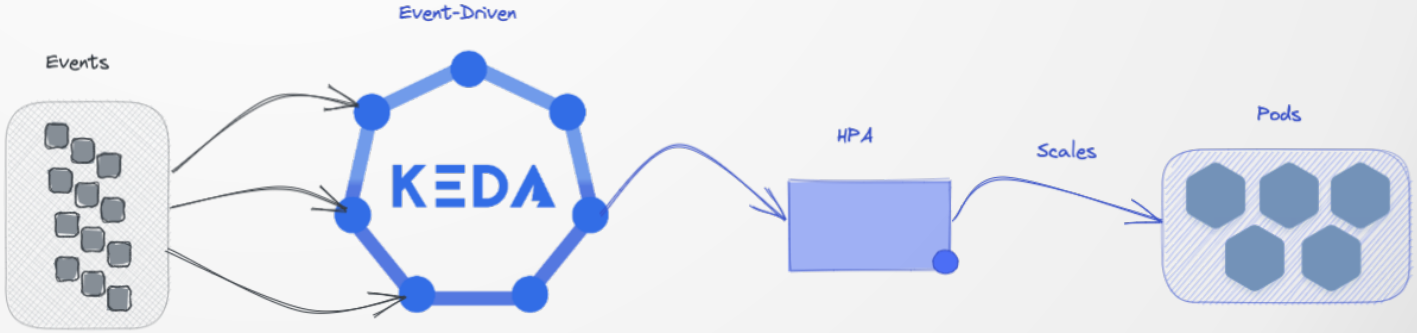
Horizonal Pod Autoscaling



KEDA - Event-Driven AutoScaler

- Scaling of Container Based of Events
- Single-Purpose and Lightweight Component
- Extend Functionality of HPA
- Built-In Scalers

https://keda.sh/



Watchdog Big Data Applications on Kubernetes Intelligently

Monitoring Big Data Applications on Kubernetes using Best of Breed Tools for Storing and Analyzing Log Data

With prometheus we scrap metrics from big data application
Important to see in yamls the metric expose configuration

Prometheus

Grafana

Grafana we create dashboard to interpreter the prometheus metrics outputs

FileBeat connect into var/logs of the pods and insert into elasticsearch all log text files

FileBeat

ElasticSearch

Kibana

In Kibana we explore the logs
create dashboards connect into elasticsearch

ElasticSearch stores all log data that filebeat inserts
The best textual search database in the market

Chaos Testing on Kubernetes

Creating Capability to Continuously but Randomly Inject Failures on Environments

Chaos engineering is the process to test distributed systems
the objective is to guarantee that he can still work properly during a disaster situation
Distributed systems are made to fail, so is time to put them to proof

Chaos Kube

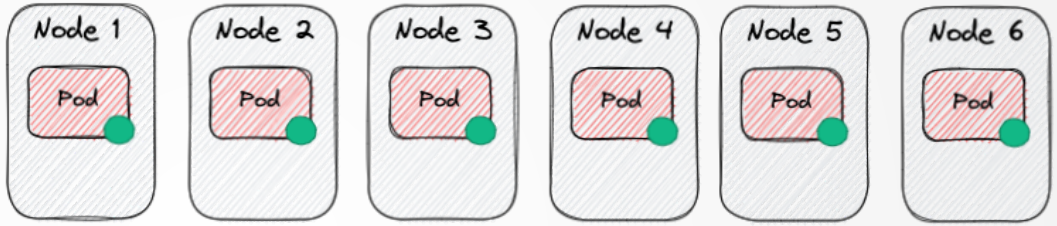
- Periodically Kills Random Pods
- Filter Targets
- Limit Specifications
- Use of Flags

https://github.com/linki/chaoskube

De-Facto Operating System
for Cloud Native Products

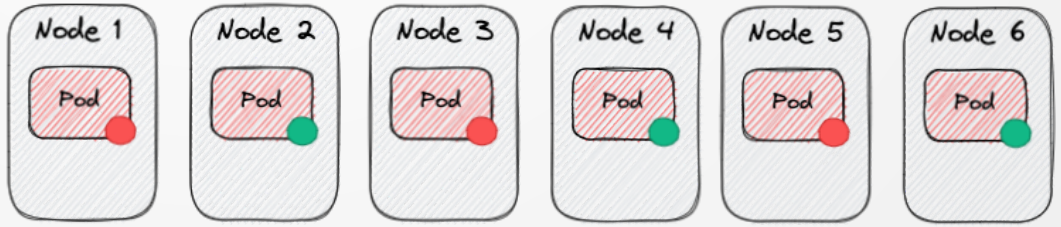
IaaS, PaaS and SaaS Softwares
for Easy and Fast Development

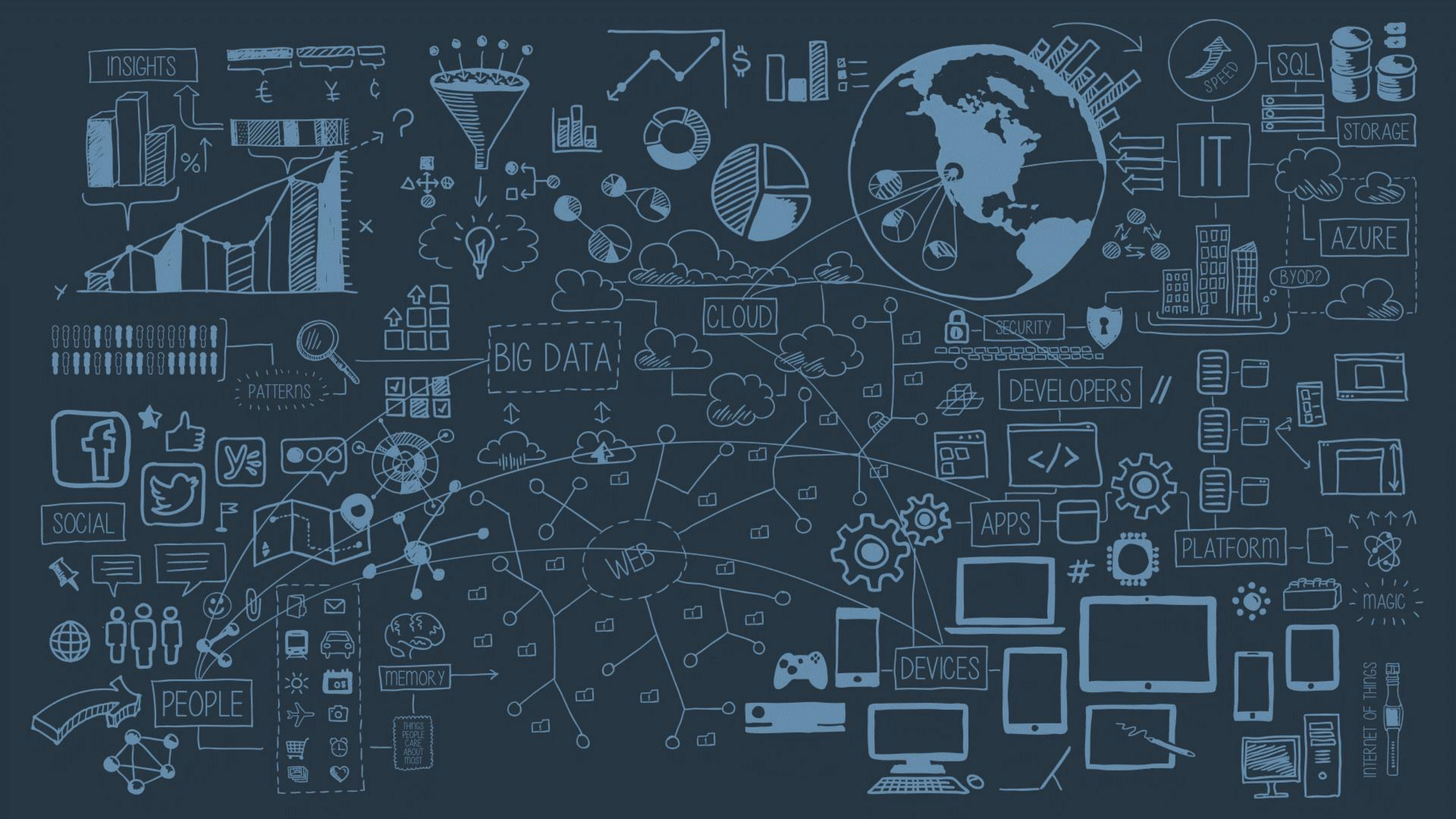
health environment



Chaos engineering work as a application
need access as cluster-admin
and start to restart pods randomly

under test environment

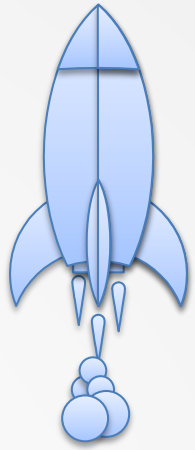



INSIGHTS
SPEED
SQL
STORAGE
IT
AZURE
BYOD?
CLOUD
SECURITY
BIG DATA
PATTERNS
DEVELOPERS
SOCIAL
APPS
PLATFORM
WEB
MAGIC
PEOPLE
MEMORY
THINGS PEOPLE CARE ABOUT MOST
DEVICES
INTERNET OF THINGS

Apache Pulsar for Pub/Sub and Real-Time Workloads

Cloud-Native, Distributed Messaging and
Streaming Platform Originally Created at Yahoo

- Originally Created at **Yahoo**
- Publish & Subscribe Pattern using Topics for Event Retention
- Cloud-Native & Distributed Messaging & Streaming Platform
- Top-Level Apache Software Foundation Project
- Local, Docker & Kubernetes Installation



- Native Support for Multiple Clusters in a Pulsar Instance
- Seamless **Geo-Replication** of Messages ~ Clusters
- Topic **Compaction** Period
- Low Publish & End-to-End Latency
- **Bindings** for Java, Go & Python
- Multiple Subscription Modes ~ **Exclusive, Shared & Failover**
- Persistent Message Storage Provided **By Apache BookKeeper**
- Light-Weight Computing Framework ~ **Pulsar Functions**
- Serverless Connector Framework ~ **Pulsar IO**
- **Tiered Storage** for Hot, Warm, Cold & Long-Term Retention
- Deduplication & Effectively-Once Semantics ~ **EOS**
- **Pulsar Schema Registry** for Type Safety ~ Client & Server Side
- Transactions (TXN) ~ Consume, Process and Produce Atomically

Producer

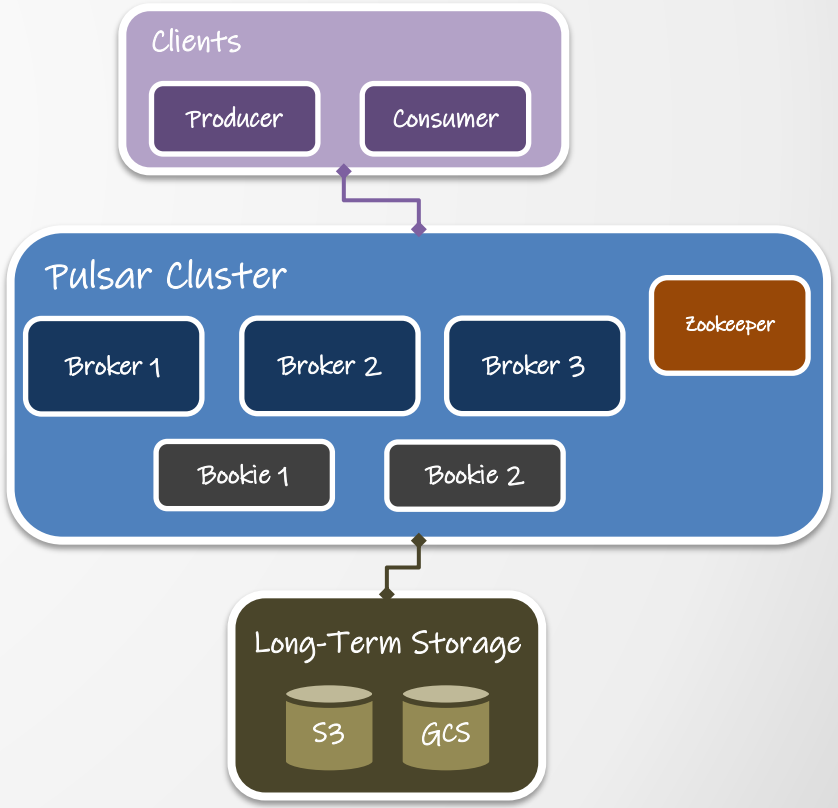
- Send Events and Publish ~ Topic in a Pulsar Broker
- Producer Idempotency
- Send Modes = Sync and Async
- **Access Mode** = Shared & Exclusive
- Compression, Batching, Chunking

Consumer

- Subscribe in a Topic via Subscription
- Receive Modes = Sync and Async
- **Positive** Acknowledgment = Individually & Cumulatively
- **Negative** Acknowledgment = Out of Order Consume
- Timeout, Dead Letter Topic, Retry
- Subscription = Exclusive, Shared, Failover & Key Shared
- Delayed Message Delivery

Topics

- Topics = Channels for Transmitting Messages
- Persistent and Non-Persistent Topics
- **Routing Modes** = Round Robin, Single, Custom
- Immediately Delete Consumed Messages
- Backlog Unacknowledged Messages
- Message Deduplication Feature = EOS
- **Namespace** = Logical Nomenclature within a Tenant



IO

- Connect with External Systems
- Source & Sink Types of Connector
- MySQL, Postgres, MongoDB, RabbitMQ
- Cassandra, ElasticSearch, Apache Kafka
- **Processing Guarantees** ~ At-Most, At-Least & Effectively Once

Functions

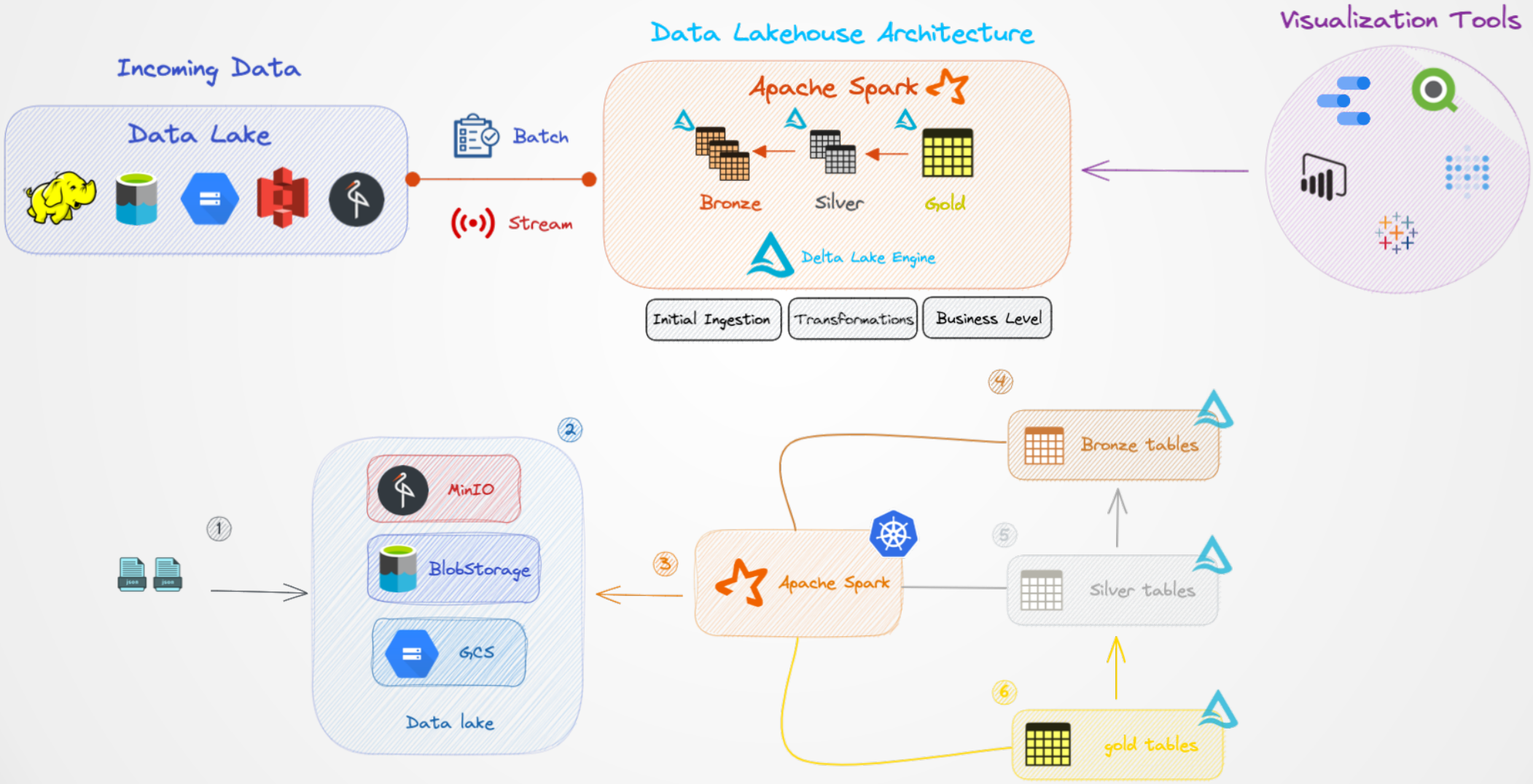
- Lightweight Compute Process
- **Consume, Apply Logic & Publish Results ~ Topic**
- Without Neighboring System Problem
- Operational Simplicity ~ Without External Processing
- Inspired By Storm, Flink, Lambda, Cloud & Azure Functions
- **Processing Guarantees** ~ At-Most, At-Least & Effectively Once

SQL

- **Query using Trino**
- Store Structured Data in Pulsar
- Connector Enables Multiple Workers
- **Data Stored** in Apache BookKeeper

Data Lakehouse Architecture using Apache Spark Engine

New Open Data Management Architecture using Flexibility, Cost-Efficiency, and Scale of Data Lakes with ACID Transactions



Modern Data Integration using Airbyte

Open-Source Data Integration for Modern Data Teams

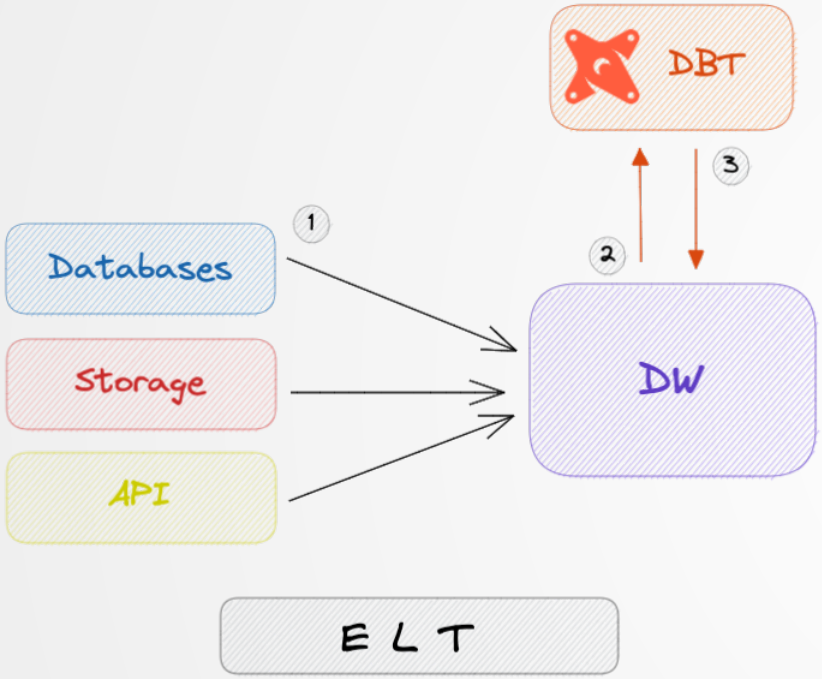
Airbyte

- Unify Data Integration Pipelines
- Connectors
- Full, Incremental and CDC
- Kubernetes Ready

https://airbyte.io/

Shipping and Munging Data Faster using DBT

Transformation Workload Allows Teams Quickly and Collaboratively Deploy Analytics Code Following Software Engineering Best Practices using SQL



DBT

- Version Control and CI/CD
- Test and Document Pipeline
- Develop using Modular SQL
- MDW as Source of Truth

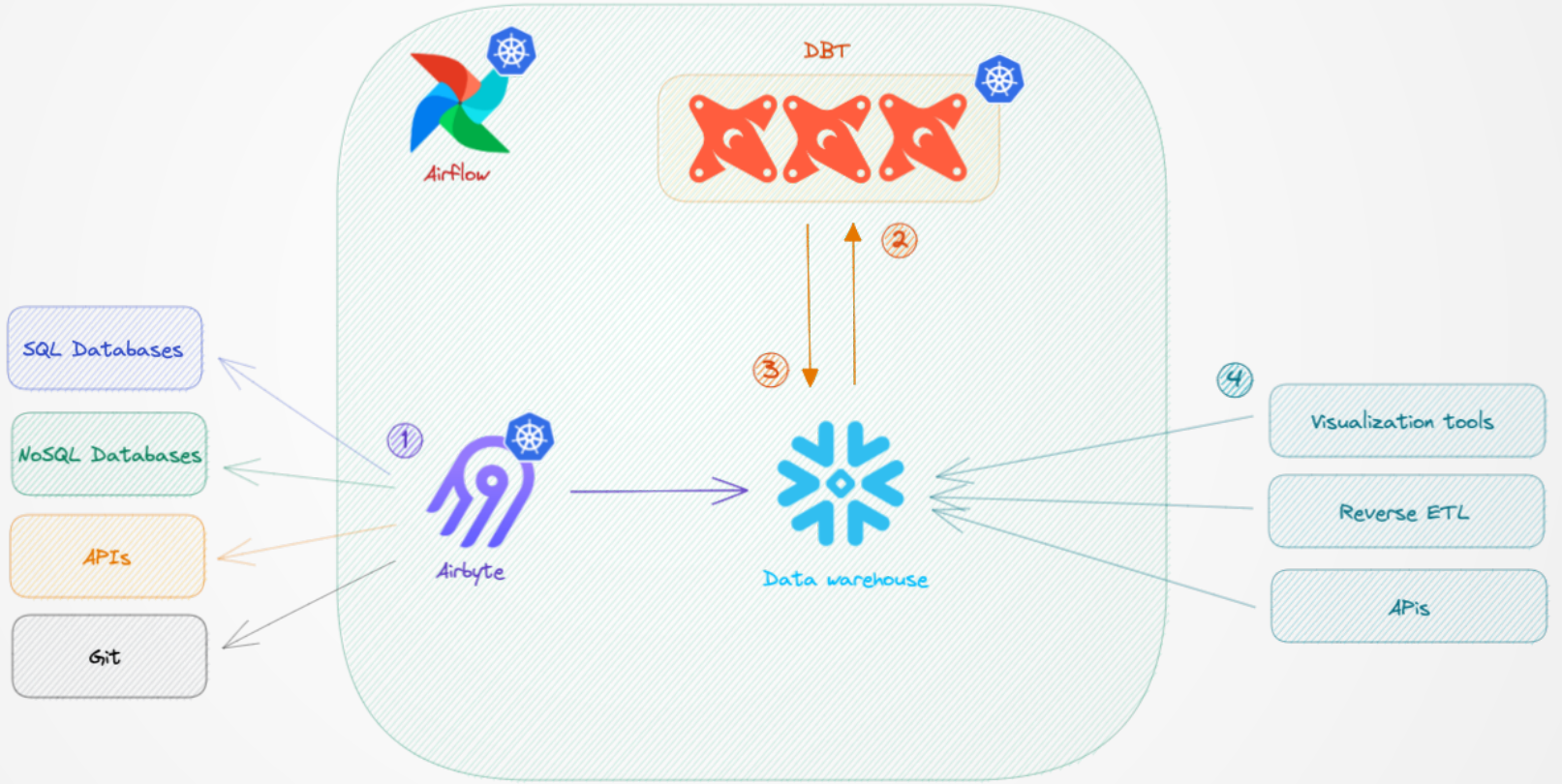
https://www.getdbt.com/

1. Data is extracted from datasources and store in data warehouse
2. Data is read by DBT and processed
3. data processed is recorded into the Data Warehouse

Airbyte, Airflow and DBT for Bullet Proof Data Pipelines

Using Airbyte for Ingestion, Airflow for Scheduling and Triggering and DBT to Perform Data Transformations using SQL Programming Language

1. Airflow trigger a Airbyte job to ingest data inside Data warehouse
2. Airflow DBT task is call after data is load and start the processing
3. Airflow DBT task is call and load the processed data into Data warehouse
4. Visualization tools, another integration process and APIs read from Data warehouse processed data



Extract | Load | Transform

ELT = Extract, Load and transform data
Give us more flexibility over the pipeline


The first step to getting anywhere is deciding you're no longer willing to stay where you are.
Anonymous
quotefancy

Lessons Learned by Working with Big Data on Kubernetes

Lessons from the Trenches about Deploying and Working
with Customers using Big Data on Kubernetes

Mindset change
Not only technical but cultural

Entire world is using
Big companies already in this path

Read a lot
many things are new so read a lot of documentation,
expend weeks if needed

Understand yaml structure
everything in kubernetes is Yaml files, so every
Yaml have a sets of features to customize

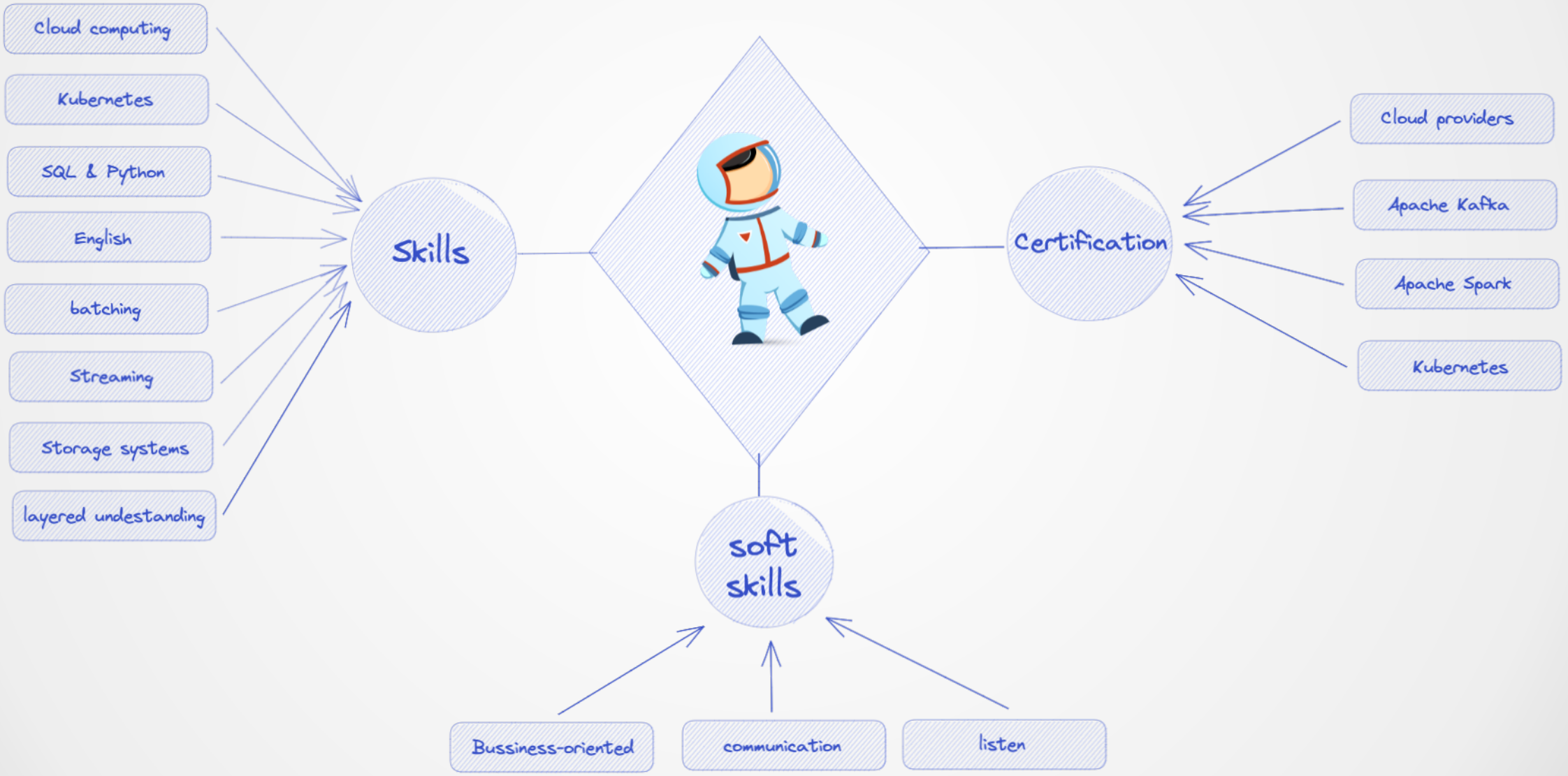
Ask help in the community
DoKC community is amazing channel to look for specialist which
are using data on kubernetes

Prepare the right pitch
We are very excited to work in kubernetes, but to make
other see that is not easy, always bring cost to the table

After 2 years working with Kubernetes
Is a wonderful environment to work, but is not
so easy to learn so keep studying

Career Path for Quantum Data Engineer

New Breed of Data Engineer, Capable of Understand Cloud and Open-Source Products and Recommend Best Data Pipeline Based on Collection Gathering Process




Live as if you were to die tomorrow. Learn as if you were to live forever.
Mahatma Gandhi
quotefancy

Obrigado!

Somos gratos pela sua participação!

engenharia de dados