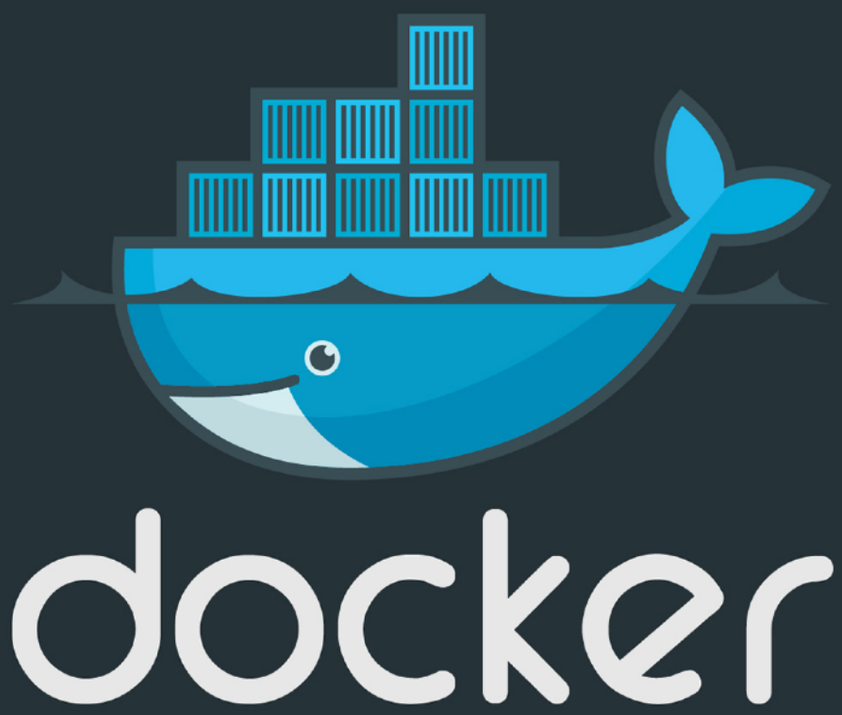
Metatron Cloud

Jinchul, SK Telecom

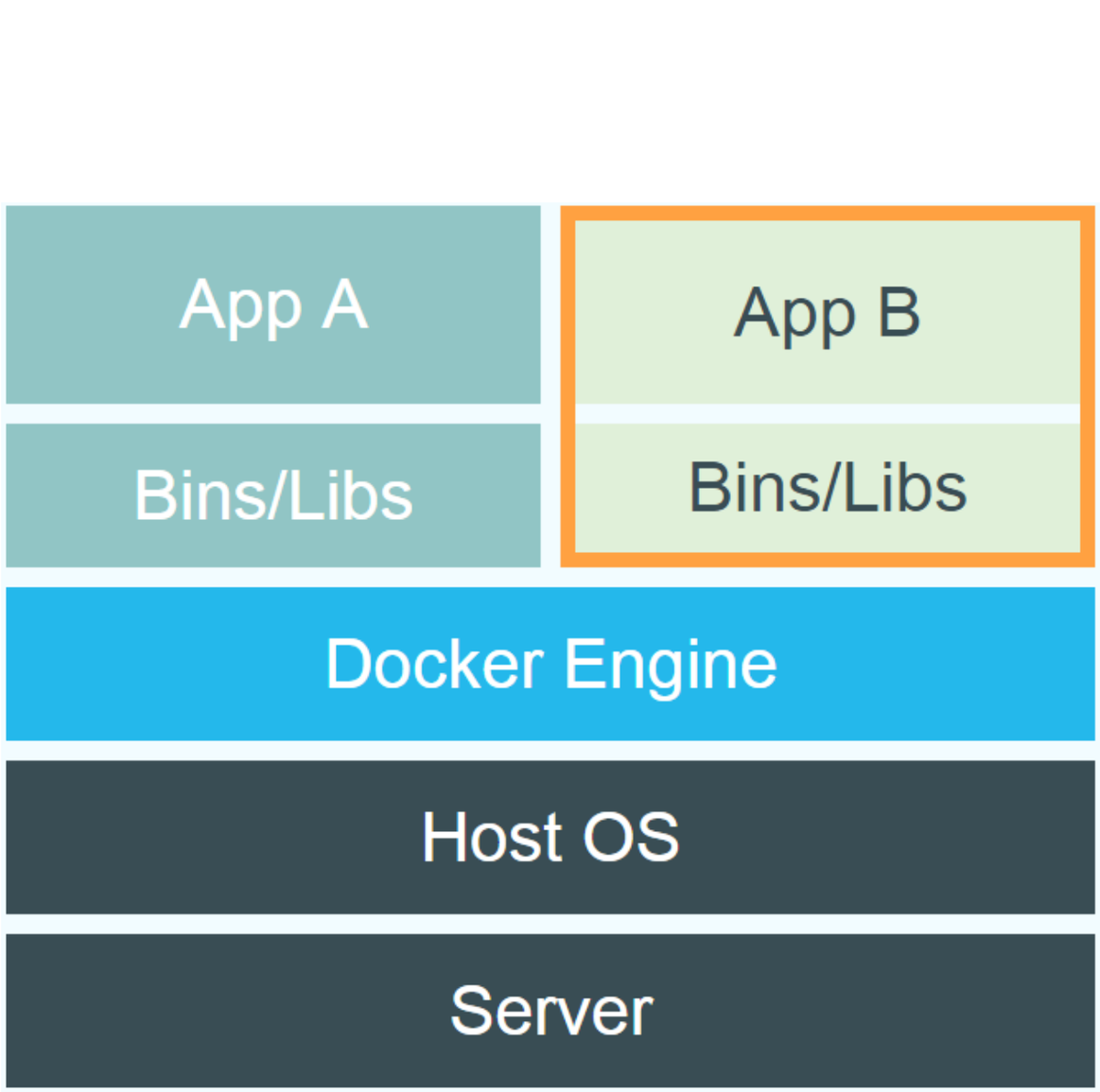
Background

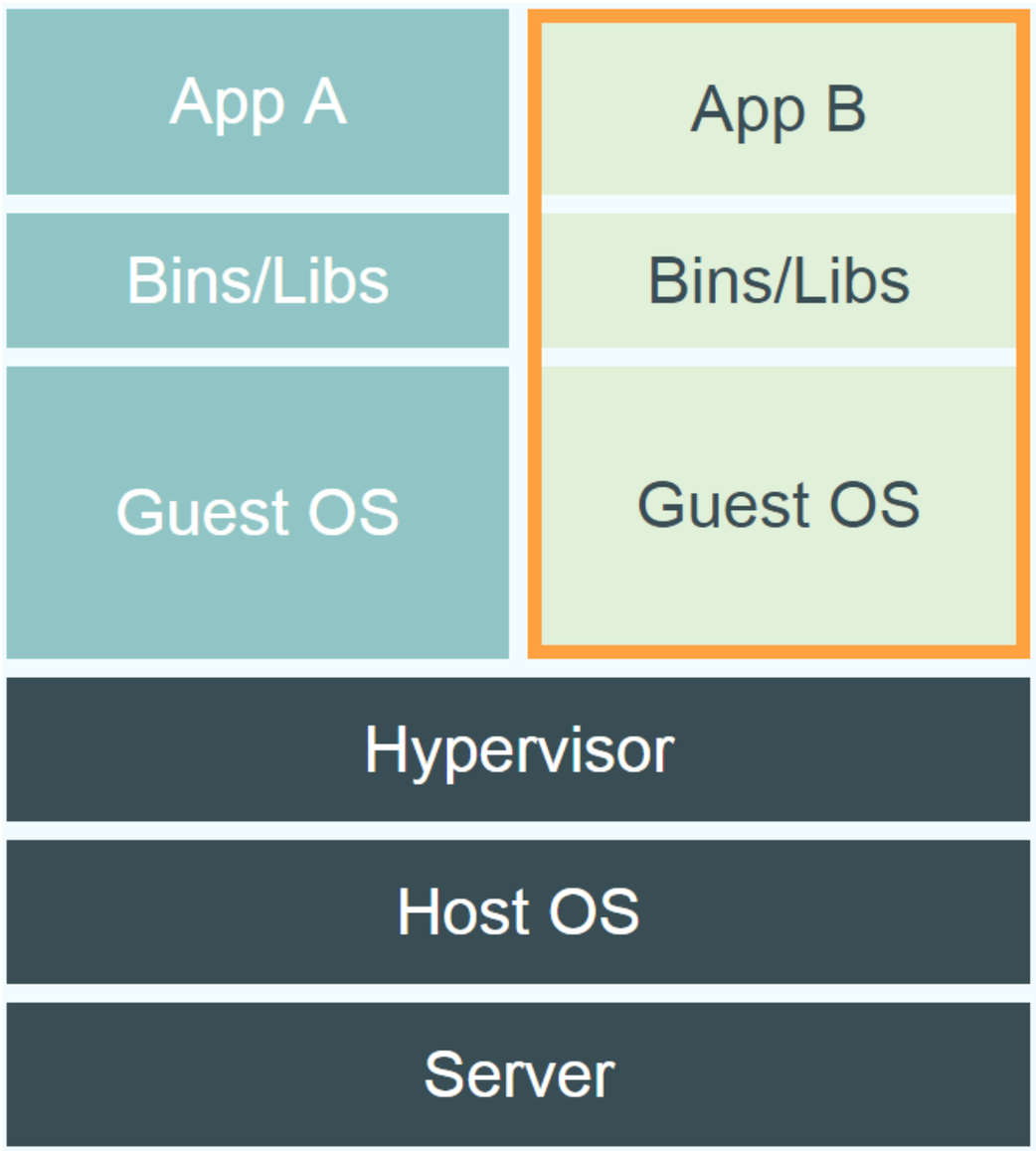


WHAT HAS DOCKER DONE FOR US?

- Continuous delivery
  - Deliver software more often and with less errors
  - No time spent on dev-to-ops handoffs
- Improved Security
  - Containers help isolate each part of your system and provides better control of each component of your system
- Run anything, anywhere
  - All languages, all databases, all operating systems
  - Any distribution, any cloud, any machine
- Reproducibility
  - Reduces the times we say “it worked on my machine”

Container vs. VM





Source: https://www.docker.com/whatisdocker/

WHAT DOES KUBERNETES DO?

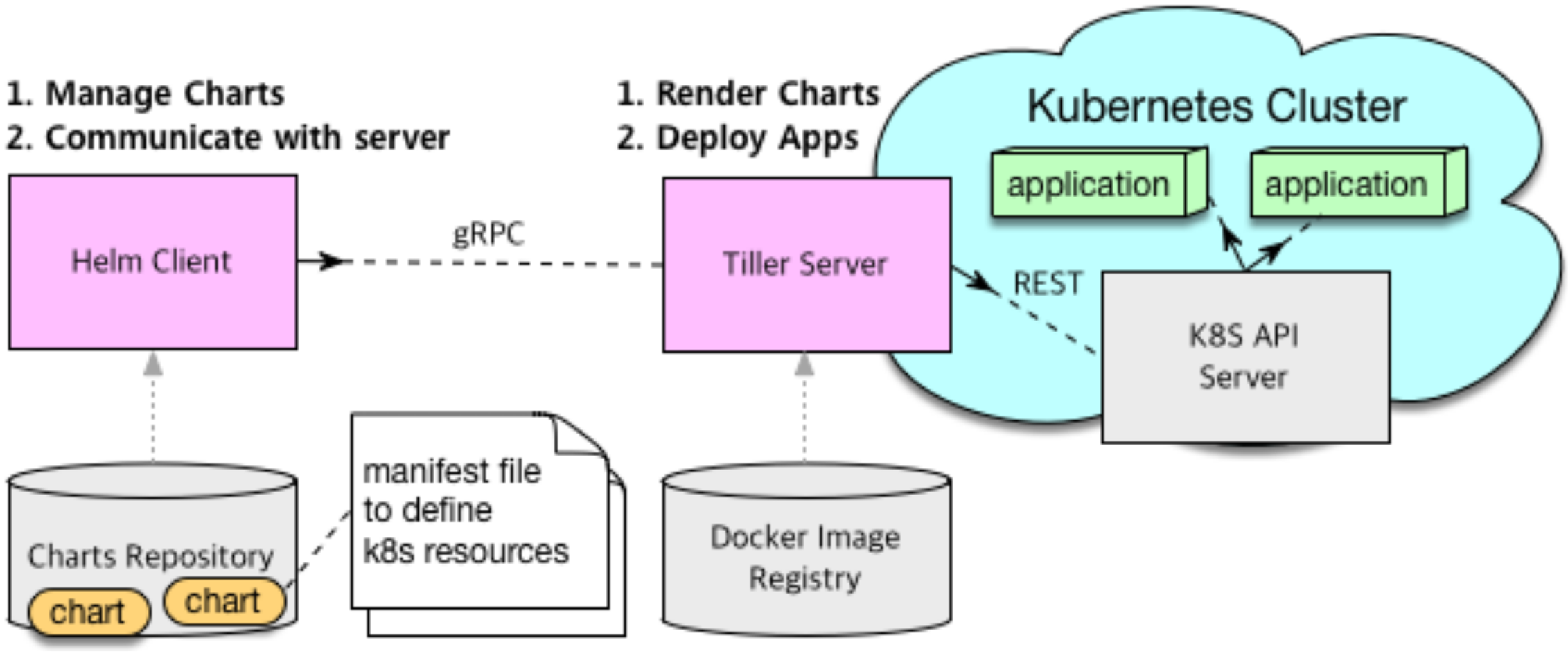
- Kubernetes is an open-source system for **automating deployment**, **scaling**, and **management** of containerized applications.
- Improves **reliability**
  - Continuously monitors and manages your containers
  - Will scale your application to handle changes in load
- Better use of **infrastructure resources**
  - Helps reduce infrastructure requirements by gracefully scaling up and down your entire platform
- Coordinates what containers run where and when across your system
- How do all the different types of containers in a system talk to each other?
- Easily coordinate deployments of your system
  - Which containers need to be deployed
  - Where should the containers be deployed

Helm Chart

Helm Architecture

Helm

- (Official) package manager for managing Kubernetes applications.
- Helm Charts helps you define, install, and upgrade Kubernetes application.
- Renders k8s manifest files and send them to k8s API => launch apps into the k8s cluster.



* **The basic chart format** consists of the templates directory, values.yaml, and other files as below.

master ··· kubernetes / charts / incubator / **druid** /

Source	Description
..	
templates	
.helmignore	METATRON-535: Adds Druid Chart
Chart.yaml	METATRON-535: Adds Druid Chart
delete.sh	METATRON-536: add scripts for druid
install.sh	METATRON-536: add scripts for druid
README.md	METATRON-535: Adds Druid Chart
values.yaml	METATRON-536: add liveness and readiness features to monitor druid

master ··· kubernetes / charts / incubator / druid / **templates** /

Source	Description
_helpers.tpl	METATRON-535: Adds Druid Chart
broker-headless-service.yaml	METATRON-536: expose service ports
broker-service.yaml	METATRON-536: fix not found hostname issue from kube-dns
broker-statefulset.yaml	METATRON-536: fix not found hostname issue from kube-dns
coordinator-headless-service.yaml	METATRON-536: expose service ports
coordinator-service.yaml	METATRON-536: fix not found hostname issue from kube-dns
coordinator-statefulset.yaml	METATRON-536: fix not found hostname issue from kube-dns
historical-headless-service.yaml	METATRON-536: expose service ports
historical-service.yaml	METATRON-536: fix not found hostname issue from kube-dns
historical-statefulset.yaml	METATRON-536: fix not found hostname issue from kube-dns
middlemanager-headless-service.yaml	METATRON-536: expose service ports
middlemanager-service.yaml	METATRON-536: fix not found hostname issue from kube-dns
middlemanager-statefulset.yaml	METATRON-536: fix not found hostname issue from kube-dns
NOTES.txt	METATRON-535: Adds Druid Chart
overlord-headless-service.yaml	METATRON-536: expose service ports
overlord-service.yaml	METATRON-536: fix not found hostname issue from kube-dns
overlord-statefulset.yaml	METATRON-536: fix not found hostname issue from kube-dns

Internal of Druid Chart

Manifest of Druid Broker in K8S

```
apiVersion: apps/v1beta1
kind: StatefulSet
...
spec:
  serviceName: {{ template "druid.fullname" . }}-broker
  replicas: {{ .Values.server.config.broker.replicaCount }}
  ...
  spec:
     containers:
      - name: {{ template "druid.name" . }}-broker
        imagePullPolicy: {{ .Values.image.pullPolicy }}
        image: "{{ .Values.image.repository }}:{{ .Values.image.tag }}"
        ports:
        - name: http-coord
          containerPort: {{ .Values.server.config.broker.port }}
          protocol: TCP
livenessProbe:
  exec:
    command:
    - /usr/local/druid/scripts/check-liveness.sh
  initialDelaySeconds: 30
  timeoutSeconds: 5
readinessProbe:
  httpGet:
    path: /status
    port: http-coord
env:
- name : HADOOP_HOSTNAME
  value: "{{ .Values.hadoop.hostname }}"
- name : DRUID_TARGET_NODE
  value: "broker"
- name : DRUID_OPTIONS
  value: -server
         ...
```

```
        volumeMounts:
        - name: datadir
          mountPath: /var/lib/druid
          subPath: data
        volumeMounts:
        - name: logdir
          mountPath: /var/logs/druid
        …
volumeClaimTemplates:
- metadata:
    name: datadir
  spec:
    accessModes: [ "ReadWriteOnce" ]
    resources:
      requests:
        storage: {{ .Values.persistence.size }}
        {{- if .Values.persistence.storageClass }}
        {{- if (eq "-" .Values.persistence.storageClass) }}
        storageClassName: ""
        {{- else }}
        storageClassName: "{{ .Values.persistence.storageClass }}"
        {{- end }}
        {{- end }}
- metadata:
    name: logdir
  spec:
    accessModes: [ "ReadWriteOnce" ]
    resources:
      requests:
        storage: {{ .Values.persistence.size }}
    {{- if .Values.persistence.storageClass }}
    {{- if (eq "-" .Values.persistence.storageClass) }}
    storageClassName: ""
    {{- else }}
    storageClassName: "{{ .Values.persistence.storageClass }}"
    {{- end }}
    {{- end }}
```

Deployment Vs. StatefulSet

```
apiVersion: apps/v1beta1
kind: StatefulSet
...
spec:
  serviceName: {{ template "druid.fullname" . }}-broker
  replicas: {{ .Values.server.config.broker.replicaCount }}
  ...
  spec:
     containers:
  - name: {{ template "druid.name" . }}-broker
    imagePullPolicy: {{ .Values.image.pullPolicy }}
    image: "{{ .Values.image.repository }}:{{ .Values.image.tag }}"
```

Liveness and Readiness Probe

```
livenessProbe:
  exec:
    command:
    - /usr/local/druid/scripts/check-liveness.sh
  initialDelaySeconds: 30
  timeoutSeconds: 5
readinessProbe:
  httpGet:
    path: /status
    port: http-coord
```

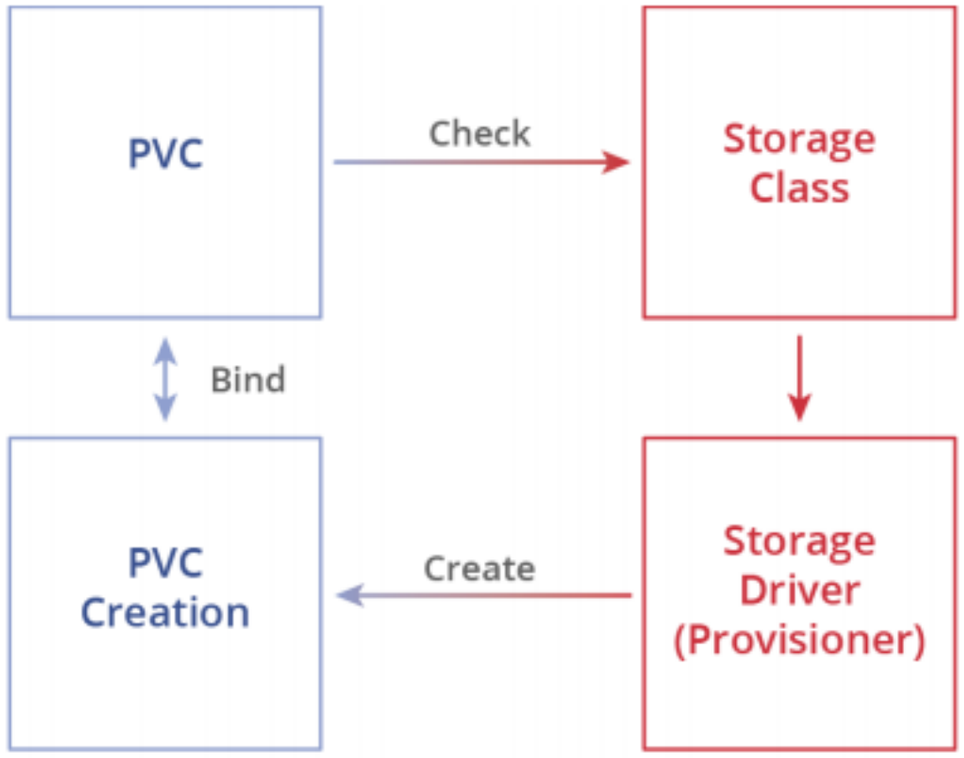
Parameter Passing

```
env:
- name : HADOOP_HOSTNAME
  value: "{{ .Values.hadoop.hostname }}"
- name : DRUID_TARGET_NODE
  value: "broker"
- name : DRUID_OPTIONS
  value: -server
         ...
         -Ddruid.metadata.storage.type={{ .Values.metadataStorage.type }}
         -Ddruid.metadata.storage.connector.connectURI={{ .Values.metadataStorage.uri }}
         -Ddruid.metadata.storage.connector.user={{ .Values.metadataStorage.user }}
         -Ddruid.metadata.storage.connector.password={{ .Values.metadataStorage.password }}
         -Ddruid.storage.type={{ .Values.storage.hadoop.type }}
         -Ddruid.storage.storageDirectory={{ .Values.storage.hadoop.directory }}
         -Ddruid.indexer.logs.type={{ .Values.indexerLogs.hadoop.type }}
         -Ddruid.indexer.logs.directory={{ .Values.indexerLogs.hadoop.directory }}
         -Ddruid.zk.service.host={{ .Values.zookeeper.hostnames }}
         …
```

Persistent Storage

```
...
      volumeMounts:
      - name: datadir
        mountPath: /var/lib/druid
        subPath: data
      volumeMounts:
      - name: logdir
        mountPath: /var/logs/druid

volumeClaimTemplates:
- metadata:
    name: datadir
  spec:
    accessModes: [ "ReadWriteOnce" ]
    resources:
      requests:
        storage: {{ .Values.persistence.size }}
    {{- if .Values.persistence.storageClass }}
    {{- if (eq "-" .Values.persistence.storageClass) }}
    storageClassName: ""
    {{- else }}
    storageClassName: "{{ .Values.persistence.storageClass }}"
    {{- end }}
    {{- end }}
  ...
```



Deploy Metatron Cloud

Ansible Script
: Deploy a Production Ready Kubernetes Cluster

Deployment via SSH

VM *or* Physical Machine

VM *or* Physical Machine

VM *or* Physical Machine

. . .

VM *or* Physical Machine

Private (SKT TACO)

Public (Azure, AWS)

Bare Metal

Kubernetes Cluster

Master HA(High Availability)

Master Node

Master Node

Master Node

Worker Node

. . .

Worker Node

VM *or* Physical Machine

VM *or* Physical Machine

VM *or* Physical Machine

. . .

VM *or* Physical Machine

Private (SKT TACO)

Public (Azure, AWS)

Bare Metal

Routing Layer

Master Node

Master Node

Master Node

API/Authentication

Data Store

Scheduler

Management

Replication

Metatron Cloud Manager

Worker Node

Worker Node

Worker Node

Worker Node

Worker Node

Worker Node

Persistent Storage

Registry

Service Layer

Private (SKT TACO)

Public (Azure, AWS)

Bare Metal

Routing Layer

Master Node
Master Node
Master Node
API/Authentication
Data Store
Scheduler
Management
Replication

Metatron Cloud Manager

Worker Node
ZK
ZK
Druid MM
Druid MM
Metatron Polaris
Druid Coordinator
Druid Broker
Druid Overlord
MariaDB

Worker Node
ZK
ZK
Druid MM
Druid MM
Metatron Polaris
Druid Historical
Druid Broker
MariaDB
Druid Overlord

Worker Node
ZK
ZK
Druid MM
Druid MM
Druid Historical
Druid Coordinator
MariaDB
MariaDB
Druid Overlord

Persistent Storage

Registry

Worker Node
ZK
ZK
Druid MM
Druid MM
Druid Historical
Druid Broker
Druid Broker
Druid Overlord
Druid Historical

Worker Node
ZK
ZK
Druid MM
Druid MM
Metatron Polaris
Druid Coordinator
Druid Broker
Druid Overlord
MariaDB

Worker Node
ZK
Druid Historical
MariaDB
Druid MM
Druid MM
Druid Overlord
Druid Coordinator
Druid Historical

Service Layer

Private (SKT TACO)

Public (Azure, AWS)

Bare Metal

Future Works