PostgreSQL HA Cluster with Query Load Balancing on Kubernetes

PGCon 2020, Ottawa
2020-05-27

SRA OSS, Inc. Japan
Bo Peng
pengbo@sraoss.co.jp

Copyright © 2020 SRA OSS, Inc. Japan All rights reserved.

Who am I?

- Bo Peng
- Organization: SRA OSS, Inc. Japan
- Experience in using PostgreSQL since 2016
- Current work
  - Open source software technical support
  - Construction work
  - Consulting
- Pgpool-II developer

Copyright © 2020 SRA OSS, Inc. Japan All rights reserved.

Outline

- Overview of Kubernetes
- Problems of running database cluster on Kubernetes
- PostgreSQL Operators on Kubernetes
- Deploy PostgreSQL HA cluster with query load balancing on Kubernetes
- Monitor PostgreSQL HA cluster on Kubernetes

Copyright © 2020 SRA OSS, Inc. Japan All rights reserved.

Overview of Kubernetes

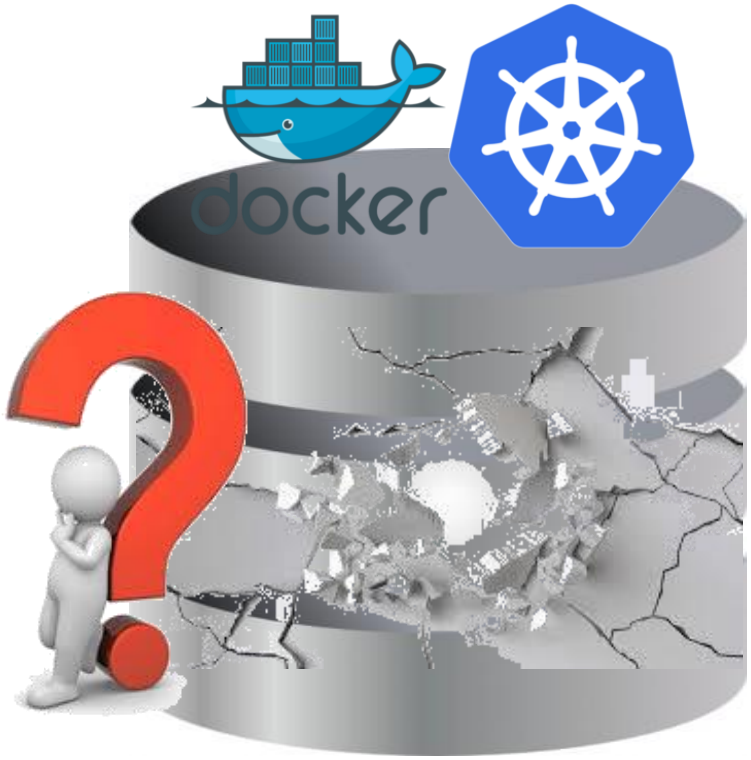
- Kubernetes is an open source container orchestration platform for automating deployment, scaling and management of application containers

- What can Kubernetes do:
  - Security, networking and storage orchestration
  - Service discovery and load balancing
    - Service is used to give a set of pods a unique IP address and DNS name
    - Traffic to Service can be load-balanced across these pods
  - Health checking and self-healing
  - Rollout and rollback
  - Automated scheduling
  - Secret and configuration management

Copyright © 2020 SRA OSS, Inc. Japan All rights reserved.

Problems of running Databases in Containers

- More and more applications are being deployed in containers on Kubernetes
- Database won't work well in containers?
  - Database is a stateful service
    - Rely on the state of the data
    - Opposite of stateless services
      - web applications, web servers
  - Persistent storage is required



Copyright © 2020 SRA OSS, Inc. Japan All rights reserved.

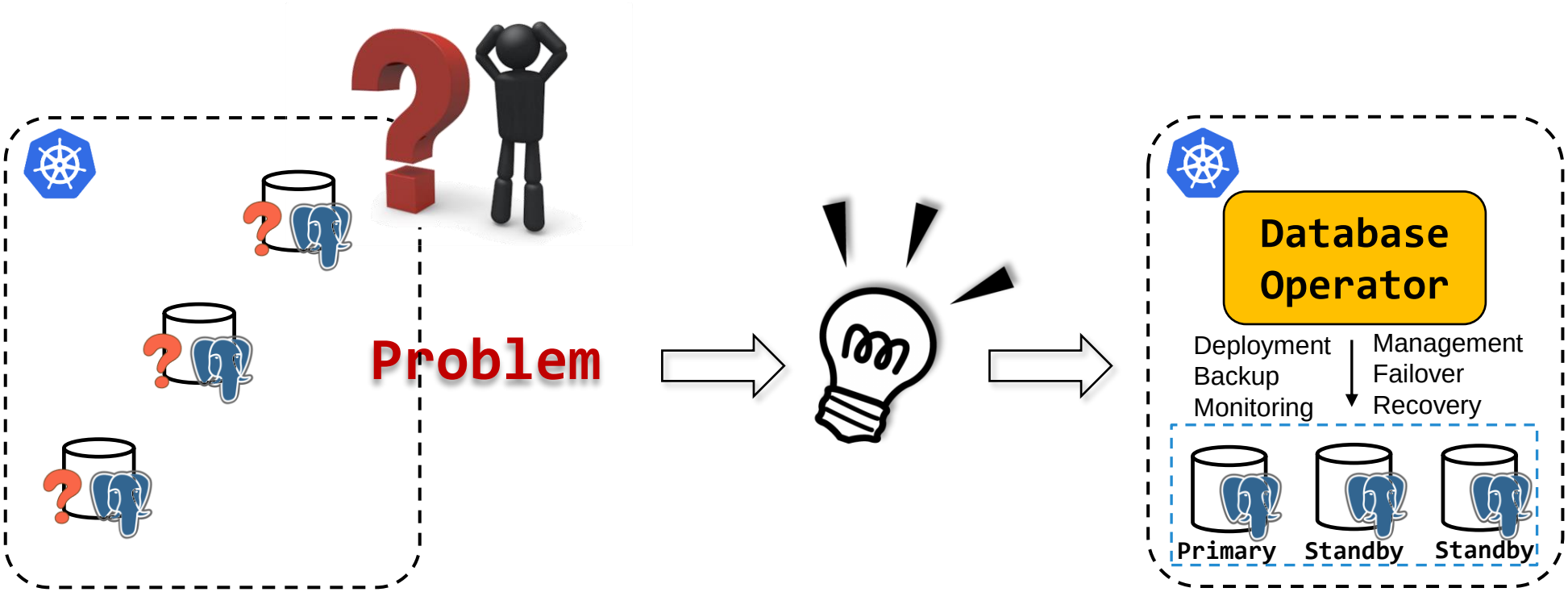
Deploy a Stateful Application using StatefulSet

- **StatefulSet**
  - Workload API object since Kubernetes 1.5
  - Manage stateful applications easily on Kubernetes
    - Manage the deployment and scaling of a set of Pods
    - Support for persistent volumes
    - Guarantee the ordering when starting and stopping the pods
    - Maintain a unique identity of these pods

Copyright © 2020 SRA OSS, Inc. Japan All rights reserved.

Problems of Running Database Cluster

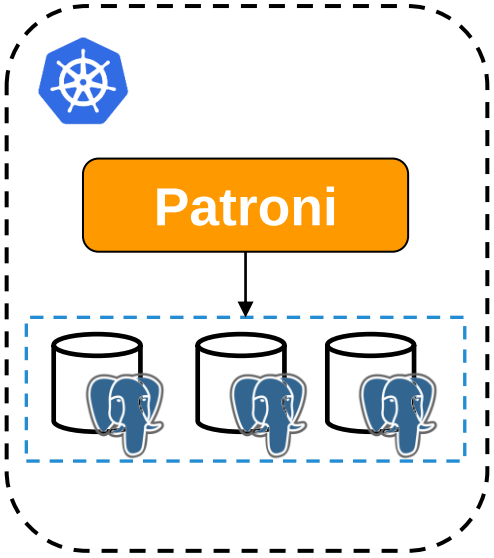
- StatefulSet provides the solution for deploying database on Kubernetes
- However, StatefulSet can't deploy and manage a database cluster
  - ? In a database cluster, replicas can't be created as easily as web servers, because you must consider the difference between Primary and Standby
  - ? How to perform automated backup, failover, recovery, etc.

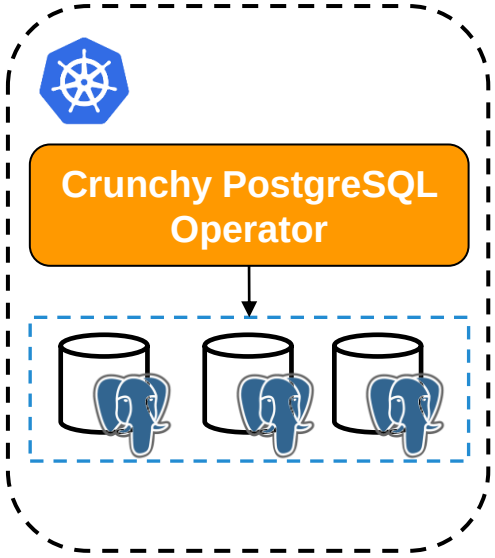


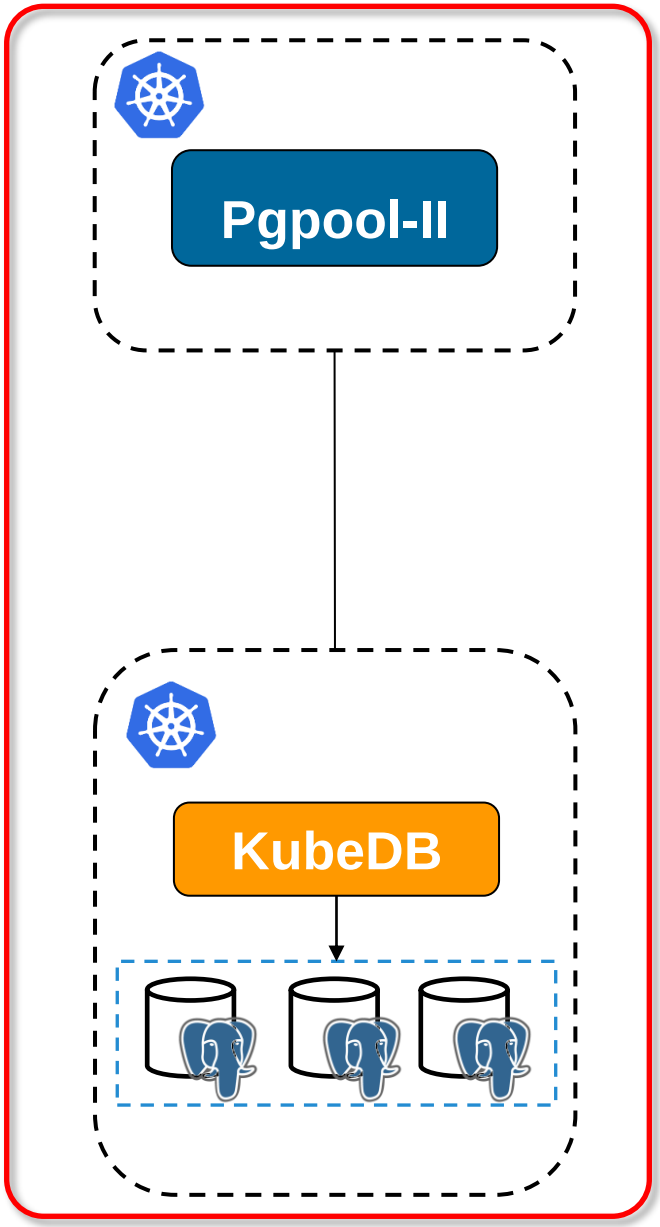
Copyright © 2020 SRA OSS, Inc. Japan All rights reserved.

PostgreSQL Operators

- What is an operator?
  - An operator is a way of deploying and managing a Kubernetes application
- Database operators simplify the processes of deploying and managing database cluster
  - Scaling, backup and restore, automatic failover, recovery, etc.
- However, PostgreSQL operators don't provide query load balancing capability. RAED/WRITE queries distribution logic is required in application source code
- To achieve query load balancing on Kubernetes, a tool which can parse the queries needs to be combined with PostgreSQL operators







Copyright © 2020 SRA OSS, Inc. Japan All rights reserved.

KubeDB

- Production quality database operator on Kubernetes

- Support for following databases
  - PostgreSQL / Elasticsearch /MySQL / MongoDB / Redis / Memcached

- Features (PostgreSQL) :
  - HA Cluster
  - Backup and restore
  - PITR from WAL archive
  - Automatic failover
  - Monitoring PostgreSQL using Prometheus

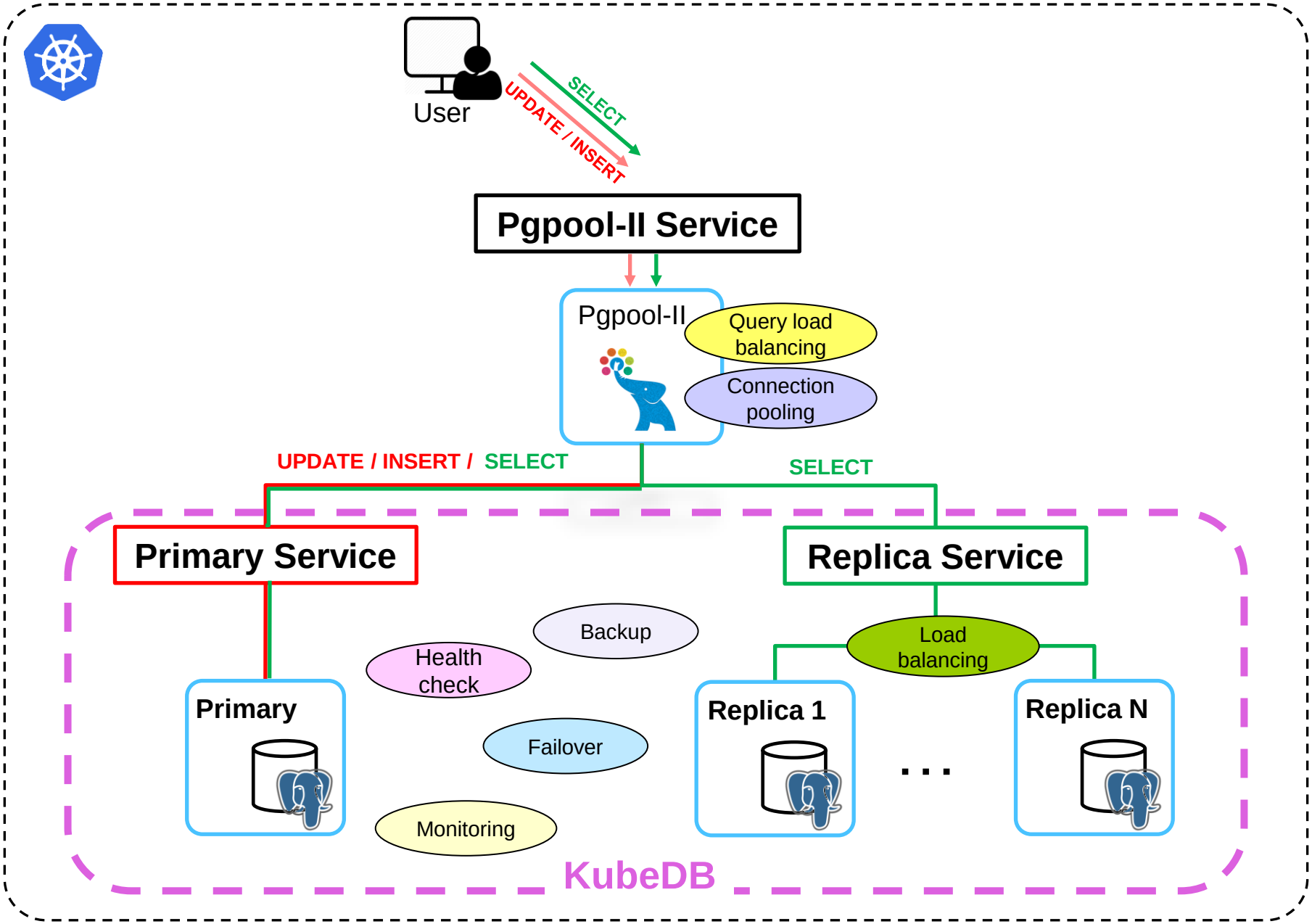
Copyright © 2020 SRA OSS, Inc. Japan All rights reserved.

Pgpool-II

- Cluster management tool for PostgreSQL
  - Support for PostgreSQL 6.4 or later
  - OSS, BSD license
- Feature-rich
  - Load balancing
  - Connection pooling
  - In memory query caching
  - Health check
  - Automatic failover
  - Watchdog (High availability for Pgpool-II)
  - Online recovery

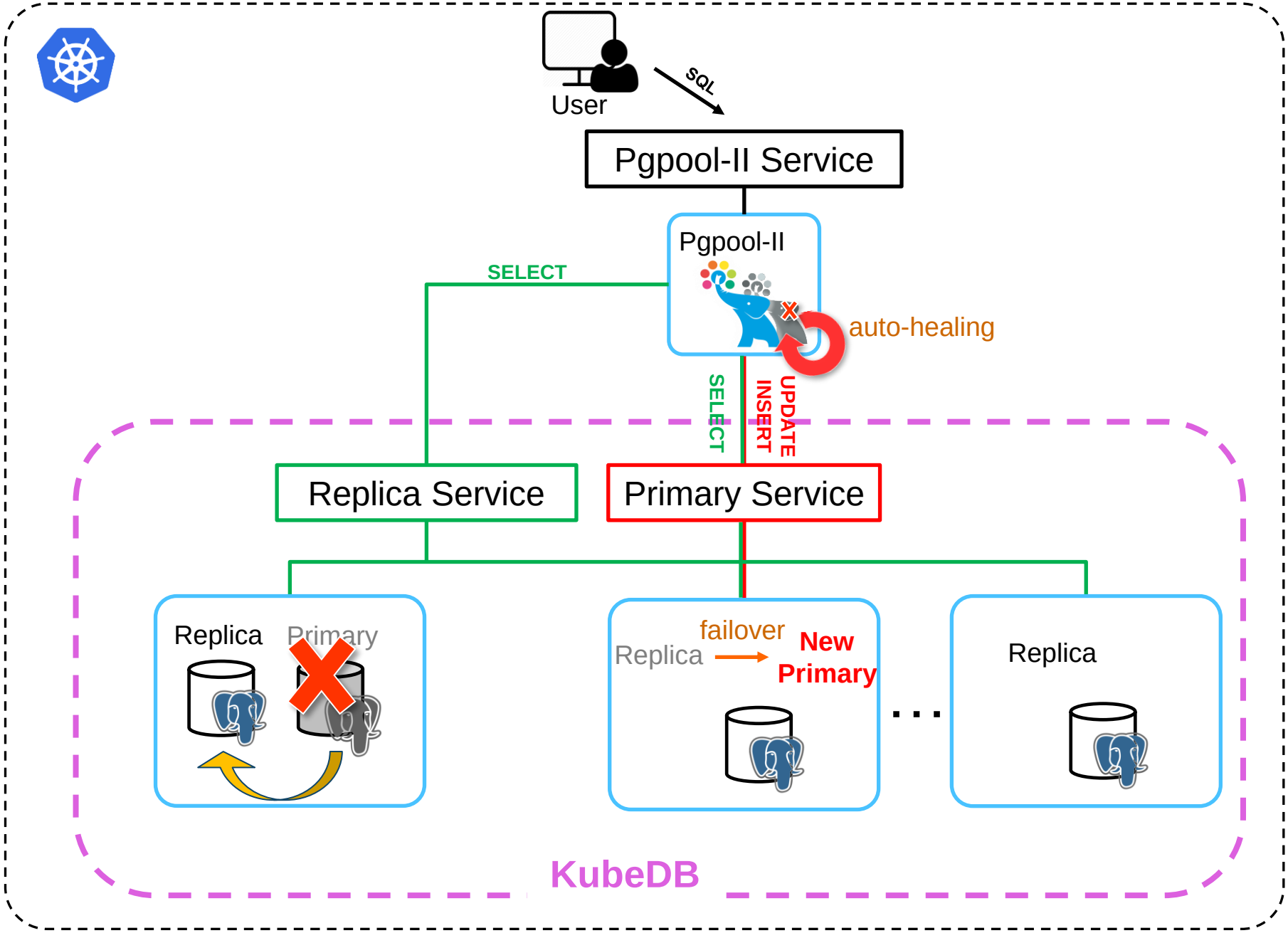
Copyright © 2020 SRA OSS, Inc. Japan All rights reserved.

Cluster Architecture with KubeDB and Pgpool-II



Copyright © 2020 SRA OSS, Inc. Japan All rights reserved.

Automatic Failover and Auto Healing



Copyright © 2020 SRA OSS, Inc. Japan All rights reserved.

How to deploy PostgreSQL HA cluster with query load balancing on Kubernetes

Copyright © 2020 SRA OSS, Inc. Japan All rights reserved.

Deploy PostgreSQL streaming replication using KubeDB

Copyright © 2020 SRA OSS, Inc. Japan All rights reserved.

Install KubeDB Operator

```
# kubectl create namespace demo
# curl -fsSL
https://raw.githubusercontent.com/kubedb/installer/v0.13.0-
rc.0/deploy/kubedb.sh | bash -s -- --namespace=demo
```

```
# kubectl get pod -n demo
NAME                                READY   STATUS    RESTARTS   AGE
kubedb-operator-5565fbdb8-g8g4v     1/1     Running   0          95s
```

Copyright © 2020 SRA OSS, Inc. Japan All rights reserved.

Install PostgreSQL SR with Hot Standby (1)

Create 3 PG servers (1 primary server, 2 hot standby servers)

```yaml
# hot-postgres.yaml
apiVersion: kubedb.com/v1alpha1
kind: Postgres
metadata:
  name: hot-postgres
spec:
  version: "11.2"
  replicas: 3
  standbyMode: Hot
  databaseSecret:
    secretName: hot-postgres-auth
  storageType: Durable
  storage:
    storageClassName: "standard"
    accessModes:
    - ReadWriteOnce
    resources:
      requests:
        storage: 2Gi
```

```yaml
# hot-postgres-auth.yaml
apiVersion: v1
kind: Secret
metadata:
  name: hot-postgres-auth
type: Opaque
data:
  POSTGRES_USER: cG9zdGdyZXM=
  POSTGRES_PASSWORD: cG9zdGdyZXM=
```

```
# echo -n 'postgres' | base64
cG9zdGdyZXM=
```

```
# kubectl apply –f hot-postgres-auth.yaml --namespace=demo
# kubectl apply –f hot-postgres.yaml --namespace=demo
```

Copyright © 2020 SRA OSS, Inc. Japan All rights reserved.

Install PostgreSQL SR with Hot Standby (2)

PostgreSQL Pods

```
# kubectl get pod -n demo --selector="kubedb.com/name=hot-postgres" --show-labels
NAME              READY   STATUS    RESTARTS   AGE   LABELS
hot-postgres-0    2/2     Running   0          15h   controller-revision--hash=hot   role=primary,
hot-postgres-1    2/2     Running   0          15h   controller-revision--hash=hot   role=replica,
hot-postgres-2    2/2     Running   0          15h   controller-revision--hash=hot   role=replica,
```

Created sequentially, in order from hot-postgres-0 to hot-postgres-2

Primary Service & Replica Service

```
# kubectl get svc -n demo
NAME                    TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
hot-postgres            ClusterIP   10.96.5.3       <none>        5432/TCP   15h
hot-postgres-replicas   ClusterIP   10.109.35.71    <none>        5432/TCP   15h
kubedb                  ClusterIP   None            <none>        <none>     15h
kubedb-operator         ClusterIP   10.107.161.44   <none>        443/TCP    15h
```

Service Name

Cluster IP

Copyright © 2020 SRA OSS, Inc. Japan All rights reserved.

Connect to PostgreSQL's Containers

Run `psql` in Primary's container

```
# kubectl exec -i -t hot-postgres-0 -n demo -- psql -U postgres
```

```
postgres=# SELECT * FROM pg_stat_replication;
pid | usesysid | username  | application_name   | client_addr |     state     |
----+----------+----------+--------------------+-------------+--------------+--
 57 |       10 | postgres | hot-postgres-1     | 10.38.0.0   |  streaming
 58 |       10 | postgres | hot-postgres-2     | 10.46.0.0   |  streaming
```

Run `psql` in Standby's container

```
# kubectl exec -i -t hot-postgres-1 -n demo -- psql -U postgres
```

```
postgres=# SELECT pg_is_in_recovery();
 pg_is_in_recovery
-------------------
 t
```

Copyright © 2020 SRA OSS, Inc. Japan All rights reserved.

Connect to PostgreSQL using Service

Connect to Primary's service

```
# psql -h 10.96.5.3 -U postgres
```

```
postgres=# SELECT * FROM pg_stat_replication;
pid | usesysid | usename  | application_name   | client_addr |     state     |
----+----------+----------+--------------------+-------------+--  -----------+-
 57 |       10 | postgres | hot-postgres-1     | 10.38.0.0   |     streaming
 58 |       10 | postgres | hot-postgres-2     | 10.46.0.0   |     streaming
```

Connect to Standby's service

```
# psql -h 10.109.35.71 -U postgres
postgres=# SELECT inet_server_addr();
 inet_server_addr
-----------------
 10.38.0.1

postgres=# ¥q
# psql -h 10.109.35.71 -U postgres
postgres=# SELECT inet_server_addr();
 inet_server_addr
-----------------
 10.46.0.2
```

Load Balance

Copyright © 2020 SRA OSS, Inc. Japan All rights reserved.

Deploy Pgpool-II on Kubernetes

Copyright © 2020 SRA OSS, Inc. Japan All rights reserved.

How to deploy Pgpool-II

Pgpool-II deployment

```yaml
# pgpool_deploy.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: pgpool
spec:
  replicas: 1
  selector:
    matchLabels:
      app: pgpool
  template:
    metadata:
      labels:
        app: pgpool
    spec:
      containers:
      - name: pgpool
        image: pgpool/pgpool:4.1
        env:
        - name: POSTGRES_USER
          valueFrom:
            secretKeyRef:
              name: hot-postgres-auth
              key: POSTGRES_USER
        - name: POSTGRES_PASSWORD
          valueFrom:
            secretKeyRef:
              name: hot-postgres-auth
              key: POSTGRES_PASSWORD
        - name: PGPOOL_PARAMS_BACKEND_HOSTNAME0
          value: "hot-postgres"
        - name: PGPOOL_PARAMS_BACKEND_HOSTNAME1
          value: "hot-postgres-replicas"
```

Create 1 Pgpool pod

Set service name to environment variables

Pgpool-II Service

```yaml
# pgpool_service.yaml
apiVersion: v1
kind: Service
metadata:
  name: pgpool
spec:
  selector:
    app: pgpool
  ports:
  - port: 9999
    protocol: TCP
```

Copyright © 2020 SRA OSS, Inc. Japan All rights reserved.

Configure Pgpool-II using Environment Variables

- Environment variables starting with "**PGPOOL_PARAMS_**" can be converted to Pgpool-II's configuration parameters and override the default configuration

Pgpool-II deployment

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: pgpool
spec:
...
    containers:
    - image: pgpool/pgpool:4.1
      name: pgpool
      env:
      ...
      - name: PGPOOL_PARAMS_BACKEND_HOSTNAME0
        value: "hot-postgres"
      - name: PGPOOL_PARAMS_BACKEND_HOSTNAME1
        value: "hot-postgres-replicas"
      - name: PGPOOL_PARAMS_NUM_INIT_CHILDREN
        value: "100"
      - name: PGPOOL_PARAMS_MAX_POOL
        value: "2"
      ...
```

Environment variables

```
# kubectl exec -i -t pgpool-696fdc6c6f-5smrl -n demo
-- env
...
HOT_POSTGRES_REPLICAS_SERVICE_HOST=10.101.206.122
HOT_POSTGRES_SERVICE_HOST=10.97.117.118
HOT_POSTGRES_SERVICE_PORT=5432
HOT_POSTGRES_REPLICAS_SERVICE_PORT=5432
PGPOOL_PARAMS_BACKEND_HOSTNAME0=hot-postgres
PGPOOL_PARAMS_BACKEND_HOSTNAME1=hot-postgres-replicas
PGPOOL_PARAMS_NUM_INIT_CHILDREN=100
PGPOOL_PARAMS_MAX_POOL=2
```

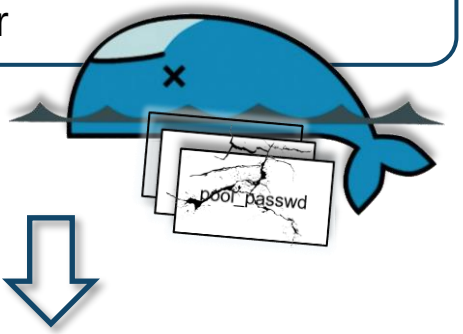
pgpool.conf

```
...
backend_port0='5432'
backend_port1='5432'
backend_hostname0='hot-postgres'
backend_hostname1='hot-postgres-replicas'
num_init_children = '100'
max_pool = '2'
```

Copyright © 2020 SRA OSS, Inc. Japan All rights reserved.

Configure Pgpool-II using ConfigMap

- Pgpool-II maintains user passwords in pool_passwd file
- If you add a new database user, you need to add the password to pool_passwd file
- However, It not easy to edit config files in container



ConfigMap

- ConfigMap API resource can store the config files
- Create a ConfigMap including config files and mount it to Pgpool-II's container
- If ConfigMap is updated, kubernetes can synchronize the mounted ConfigMap

```yaml
# pgpool_configmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: pgpool-config
  labels:
    app: pgpool-config
data:
  pgpool.conf: |-
    listen_addresses = '*'
    port = 9999
    socket_dir = '/var/run/postgresql'
    pcp_listen_addresses = '*'
    pcp_port = 9898
...
  pcp.conf: |-
    postgres:e8a48653851
...
  pool_passwd: |-
    postgres:md53175bce1
    user1:md542b72f913c3
    user2:md548b83a2a920
...
  pool_hba.conf: |-
    local   all     all
    host    all     all
    host    all     all
    host    all     all
```

Mount

```yaml
# pgpool_deploy.yaml
apiVersion: apps/v1
kind: Deployment
...
    spec:
      containers:
      - image: pgpool/pgpool:4.1
        command: ["sh", "-c",
"$PGPOOL_BINARY_DIR/pgpool -n -f
$PGPOOLCONF -F $PCP_CONF -a
$POOL_HBA_CONF"]
        env:
       ...
        volumeMounts:
        - name: pgpool-config
        mountPath: /usr/local/pgpool-
II/etc
      volumes:
      - name: pgpool-config
        configMap:
          name: pgpool-config
```

Copyright © 2020 SRA OSS, Inc. Japan All rights reserved.

Deploy Pgpool-II

```
# kubectl apply –f pgpool_service.yaml --namespace=demo
# kubectl apply –f pgpool_deploy.yaml --namespace=demo
```

```
# kubectl get pod -n demo
NAME                              READY   STATUS    RESTARTS   AGE
hot-postgres-0                    2/2     Running   0          16h
hot-postgres-1                    2/2     Running   0          16h
hot-postgres-2                    2/2     Running   0          16h
kubedb-operator-5565fbdb8-g8g4v   1/1     Running   1          16h
pgpool-696fdc6c6f-5smrl           2/2     Running   0          15h
```

```
# kubectl get svc -n demo
NAME                    TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
hot-postgres            ClusterIP   10.96.5.3       <none>        5432/TCP   16h
hot-postgres-replicas   ClusterIP   10.109.35.71    <none>        5432/TCP   16h
kubedb                  ClusterIP   None            <none>        <none>     16h
kubedb-operator         ClusterIP   10.107.161.44   <none>        443/TCP    16h
pgpool                  ClusterIP   10.103.121.95   <none>        9999/TCP   15h
```

Copyright © 2020 SRA OSS, Inc. Japan All rights reserved.

Query Load Balancing

```
# psql -h 10.103.121.95 -U postgres -p 9999 -c "show pool_nodes"
node_id|   hostname      | port |status| lb_weight |  role   | select_cnt | load_balance_node| replication_delay
-------+-----------------+------+------+----------+---------+------------+------------------+-----------------+
0      |hot-postgres        | 5432 | up   | 0.500000  | primary | 0          | false            | 0
1      |hot-postgres-replicas| 5432 | up   | 0.500000  | standby | 0          | true             | 0
```

```
# psql -h 10.103.121.95 -U postgres -p 9999 -c "SELECT 1"
  ?column?
 ----------
         1
 (1 row)
# psql -h 10.103.121.95 -U postgres -p 9999 -c "SELECT 1"
  ?column?
 ----------
         1
 (1 row)
...

# psql -h 10.103.121.95 -U postgres -p 9999 -c "show pool_nodes"
node_id |  hostname       | port |status| lb_weight|  role    | select_cnt | load_balance_node| replication_delay
--------+-----------------+------+------+----------+---------+------------+------------------+-----------------+
0       |hot-postgres        | 5432 | up    | 0.500000 | primary | 3          | false            | 0
1       |hot-postgres-replicas| 5432 | up    | 0.500000 | standby | 3          | true             | 0
```

SELECT query counts issued to each backend

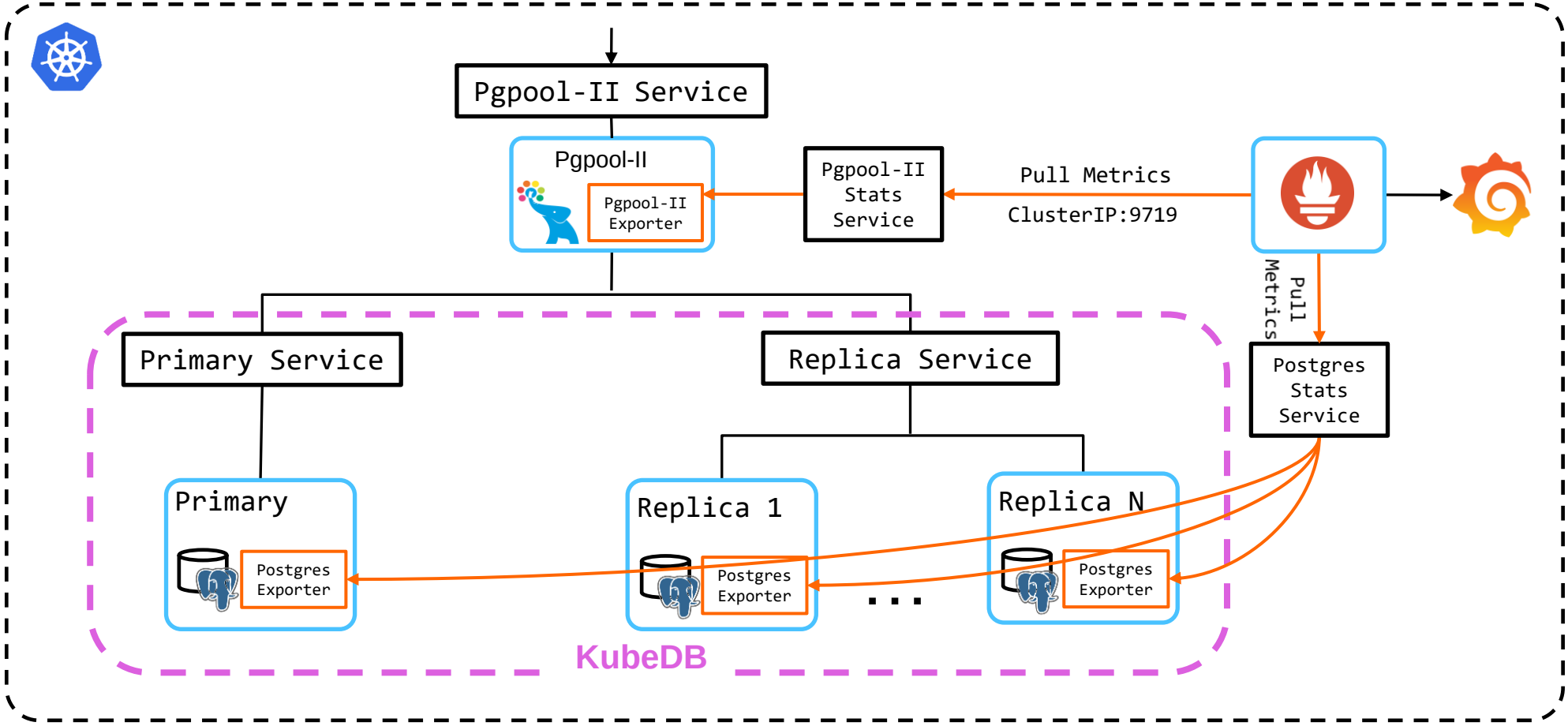
Copyright © 2020 SRA OSS, Inc. Japan All rights reserved.

Monitor PostgreSQL Cluster on Kubernetes

Copyright © 2020 SRA OSS, Inc. Japan All rights reserved.

Monitoring

- Although Kubernetes provides a way to monitor the basic cluster status, this is not sufficient for practical cluster management
- It is useful to expose application metrics and visualize them
  - Expose specific metrics via an HTTP endpoint (using Postgres Exporter and Pgpool-II Exporter)
  - Prometheus Server pulls real-time metrics and stores them in a time series database



Copyright © 2020 SRA OSS, Inc. Japan All rights reserved.

Pgpool-II Exporter

- Prometheus exporter for Pgpool-II metrics
- Implemented in Go
- https://github.com/pgpool/pgpool2_exporter
- Metrics

Name	Description
pgpool2_frontend_total	Number of total child processes
pgpool2_frontend_used	Number of used child processes
pgpool2_pool_nodes_status	Backend node status
pgpool2_pool_nodes_replication_delay	Replication delay
pgpool2_pool_nodes_select_cnt	SELECT query counts issued to each backend
pgpool2_pool_cache_cache_hit_ratio	Query cache hit ratio
pgpool2_pool_cache_num_cache_entries	Number of used cache entries
pgpool2_pool_cache_num_hash_entries	Number of total hash entries
pgpool2_pool_cache_used_hash_entries	Number of used hash entries

Copyright © 2020 SRA OSS, Inc. Japan All rights reserved.

Deploy Pgpool-II Exporter

```yaml
# pgpool_depoly.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: pgpool
spec:
  replicas: 1
  selector:
    matchLabels:
      app: pgpool
  template:
    metadata:
      labels:
        app: pgpool
    spec:
      containers:
      - name: pgpool
        image: pgpool/pgpool:4.1
   ...
      - name: pgpool-stats
        image: pgpool/pgpool2_exporter:1.0
        env:
        - name: POSTGRES_USER
          valueFrom:
            secretKeyRef:
              name: hot-postgres-auth
              key: POSTGRES_USER
        - name: POSTGRES_PASSWORD
          valueFrom:
            secretKeyRef:
              name: hot-postgres-auth
              key: POSTGRES_PASSWORD
        - name: PGPOOL_SERVICE
          value: "localhost"
```

```yaml
# pgpool2_exporter_service.yaml
apiVersion: v1
kind: Service
metadata:
  name: pgpool-stats
  annotations:
    prometheus.io/path: /metrics
    prometheus.io/port: "9719"
    prometheus.io/scrape: "true"
spec:
  selector:
    app: pgpool
  ports:
  - port: 9719
    targetPort: 9719
    protocol: TCP
```

Set annotations to allow Prometheus server to discover Pgpool-II exporter service endpoint

Deploy Pgpool-II Exporter container in Pgpool-II pod

```
# kubectl apply –f pgpool_depoly.yaml --namespace=demo
# kubectl apply –f pgpool2_exporter_service.yaml --
namespace=demo
```

Set connection destination environment variable

Copyright © 2020 SRA OSS, Inc. Japan All rights reserved.

Prometheus Scrapping Configuration

```yaml
# prometheus_configmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: prometheus-config
  labels:
    app: prometheus-demo
data:
  prometheus.yml: |-
    global:
      scrape_interval: 5s
      evaluation_interval: 5s
    scrape_configs:
 - job_name: 'pgpool'
      honor_labels: true
      scheme: http
      kubernetes_sd_configs:
      - role: endpoints
      relabel_configs:
      - source_labels: [__meta_kubernetes_service_annotation_prometheus_io_scrape,
__meta_kubernetes_service_annotation_prometheus_io_port]
        separator: ;
        regex: true;(.*)
        action: keep
      - source_labels: [__meta_kubernetes_service_annotation_prometheus_io_scheme]
        action: drop
        regex: https
      - source_labels: [__meta_kubernetes_service_name]
        separator: ;
        regex: (pgpool-stats)
        action: keep
      - source_labels: [__meta_kubernetes_service_annotation_prometheus_io_path]
        action: replace
        target_label: __metrics_path__
        regex: (.+)
      - source_labels: [__address__, __meta_kubernetes_service_annotation_prometheus_io_port]
        action: replace
        target_label: __address__
        regex: ([^:]+)(?::¥d+)?;(¥d+)
        replacement: $1:$2
      - source_labels: [__meta_kubernetes_namespace]
        separator: ;
        regex: (.*)
        target_label: namespace
        replacement: $1
        action: replace
      - source_labels: [__meta_kubernetes_service_name]
        separator: ;
        regex: (.*)
        target_label: service
        replacement: $1
        action: replace
      - action: labelmap
        regex: __meta_kubernetes_service_label_(.+)
```

- Prometheus can discover scraping targets using Kubernetes API
- Available meta labels
  - __meta_kubernetes_namespace
  - __meta_kubernetes_service_annotation_<annotationname>
  - __meta_kubernetes_service_annotationpresent_<annotationname>
  - __meta_kubernetes_service_cluster_ip
  - __meta_kubernetes_service_external_name
  - __meta_kubernetes_service_label_<labelname>
  - __meta_kubernetes_service_labelpresent_<labelname>
  - __meta_kubernetes_service_name
  - __meta_kubernetes_service_port_name
  - __meta_kubernetes_service_port_protocol
  - __meta_kubernetes_service_type

Copyright © 2020 SRA OSS, Inc. Japan All rights reserved.

Prometheus server

```yaml
# prometheus.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: prometheus
  labels:
    app: prometheus-demo
spec:
  replicas: 1
  selector:
    matchLabels:
      app: prometheus
  template:
    metadata:
      labels:
        app: prometheus
    spec:
      serviceAccountName: prometheus
      containers:
      - name: prometheus
        image: prom/prometheus:v2.5.0
        args:
        - "--config.file=/etc/prometheus/prometheus.yml"
        ports:
        - containerPort: 9090
        volumeMounts:
        - name: prometheus-config
          mountPath: /etc/prometheus/
        - name: prometheus-storage
          mountPath: /prometheus/
      volumes:
      - name: prometheus-config
        configMap:
          defaultMode: 420
          name: prometheus-config
      - name: prometheus-storage
        emptyDir: {}
```

Mount ConfigMap

```yaml
# prometheus_service.yaml
apiVersion: v1
kind: Service
metadata:
  name: prometheus
spec:
  selector:
    app: prometheus
  ports:
  - port: 9090
    protocol: TCP
```

Copyright © 2020 SRA OSS, Inc. Japan All rights reserved.

Deploy Prometheus Server

```
# kubectl create namespace monitoring
# kubectl apply -f prometheus_configmap.yaml --namespace=monitoring
# kubectl apply -f prometheus.yaml --namespace=monitoring
```

Prometheus Alerts Graph Status Help

Targets

All Unhealthy

kubedb-databases (3/3 up) show less

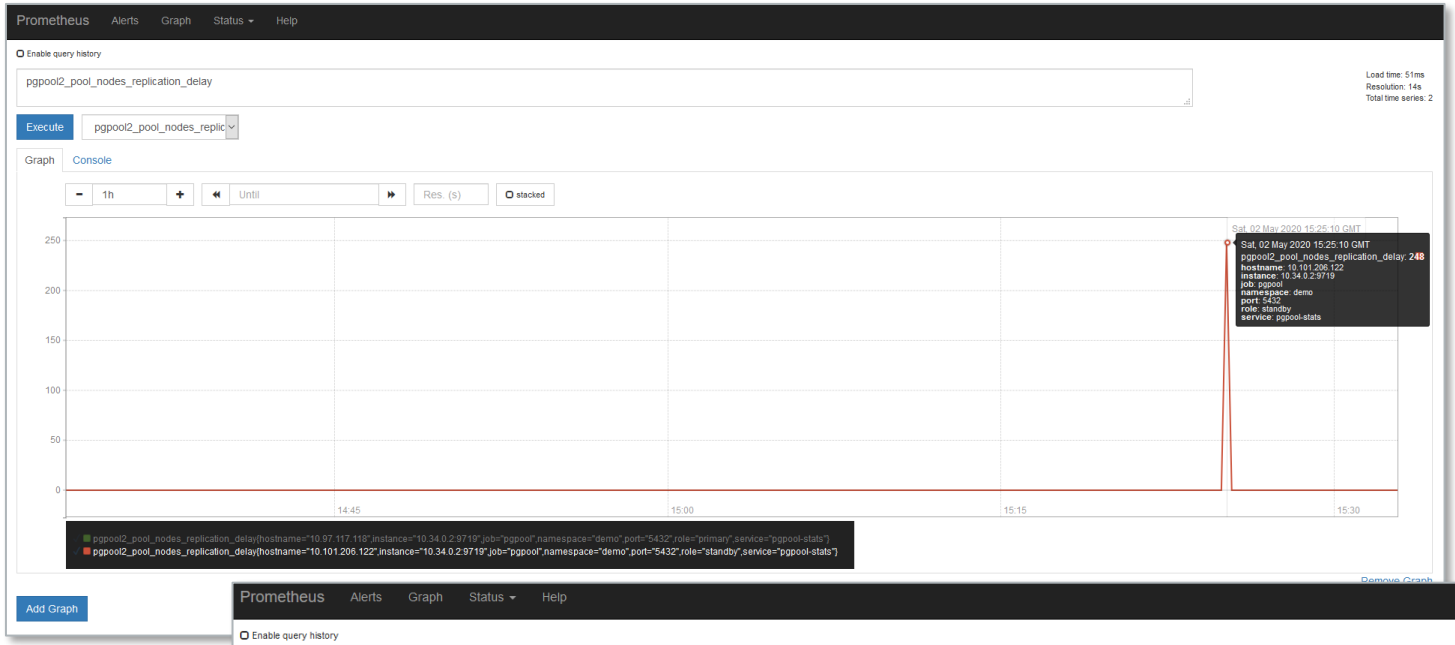
Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://10.38.0.1:56790/metrics	UP	instance="10.38.0.1:56790" job="kubedb-databases" kubedb_com_kind="Postgres" kubedb_com_name="hot-postgres" kubedb_com_role="stats" namespace="demo" service="hot-postgres-stats"	2.842s ago	68.35ms	
http://10.40.0.3:56790/metrics	UP	instance="10.40.0.3:56790" job="kubedb-databases" kubedb_com_kind="Postgres" kubedb_com_name="hot-postgres" kubedb_com_role="stats" namespace="demo" service="hot-postgres-stats"	293ms ago	42.14ms	
http://10.46.0.2:56790/metrics	UP	instance="10.46.0.2:56790" job="kubedb-databases" kubedb_com_kind="Postgres" kubedb_com_name="hot-postgres" kubedb_com_role="stats" namespace="demo" service="hot-postgres-stats"	4.252s ago	70.92ms	

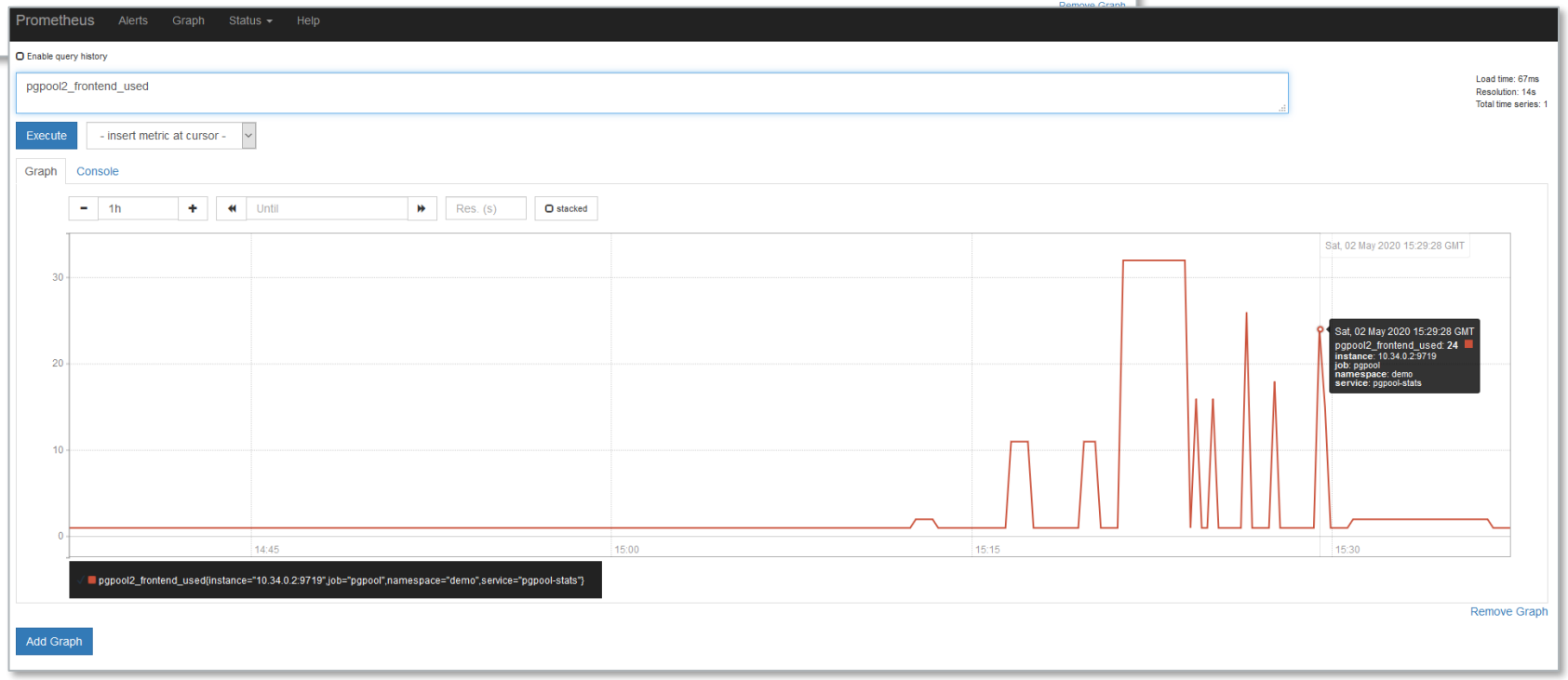
pgpool (1/1 up) show less

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://10.38.0.2:9719/metrics	UP	app="pgpool-stats" instance="10.38.0.2:9719" job="pgpool" namespace="demo" service="pgpool-stats"	4.537s ago	7.903ms	

Copyright © 2020 SRA OSS, Inc. Japan All rights reserved.

Prometheus Metrics Graph





Copyright © 2020 SRA OSS, Inc. Japan All rights reserved.

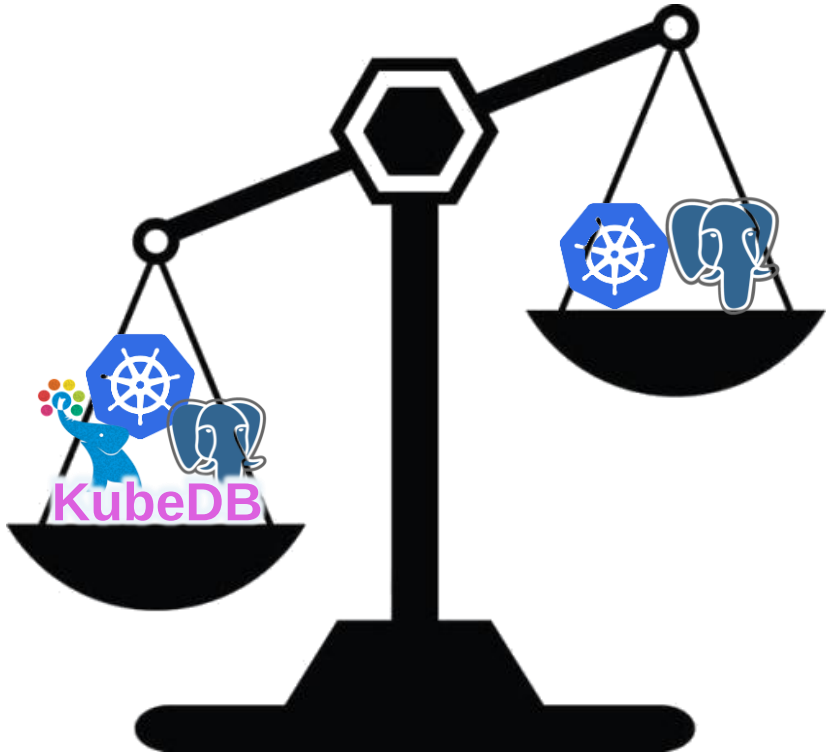
Pros and Cons of using KubeDB and Pgpool-II

Pros

- Simplify the processes of deploying and managing PostgreSQL cluster
- Feature-rich
  - Backup and restore
  - Recovery
  - Automatic failover
  - Monitoring
  - Query load balancing

Cons

- The latest version of PostgreSQL is not yet supported
- Synchronous replication is not yet supported



Copyright © 2020 SRA OSS, Inc. Japan All rights reserved.

Future Work

- Make Pgpool-II more Kubernetes-friendly
  - Think about how to set the configuration parameter "backend_application_name", because only one standby is visible to Pgpool-II
  - Enhance "SHOW …" and PCP command to output more helpful cluster state information
- Improve Pgpool-II Exporter
  - Expose more helpful metrics
  - Support for custom queries
- Collect logs from Kubernetes cluster and monitor PostgreSQL activity

Copyright © 2020 SRA OSS, Inc. Japan All rights reserved.

Summary

- Problems of running databases on Kubernetes
- PostgreSQL operators on Kubernetes
- Deploy PostgreSQL HA cluster with query load balancing on Kubernetes
  - KubeDB (PostgreSQL operator)
  - Pgpool-II (Query load balancing)
- Monitoring
  - Prometheus
  - Postgres exporter
  - Pgpool-II exporter

Copyright © 2020 SRA OSS, Inc. Japan All rights reserved.

Appendix

- Config files for building KubeDB and Pgpool-II on Kubernetes
  - https://github.com/pgpool/pgpool2_on_k8s
- Pgpool-II
  - https://pgpool.net/
- KubeDB
  - https://kubedb.com/
- Pgpool-II Docker images for Kubernetes
  - https://hub.docker.com/repository/docker/pgpool/pgpool
  - https://hub.docker.com/repository/docker/pgpool/pgpool2_exporter

Copyright © 2020 SRA OSS, Inc. Japan All rights reserved.

Thank you!

Copyright © 2020 SRA OSS, Inc. Japan All rights reserved.