DBaaS: An autonomous journey in a multi-cloud Kubernetes environment

MariaDB

SCAN ME

About me:

{

Title: "Staff Engineer",
Profile: "Golang",
Env: "Multi-Cloud",
Platform: "Kubernetes"

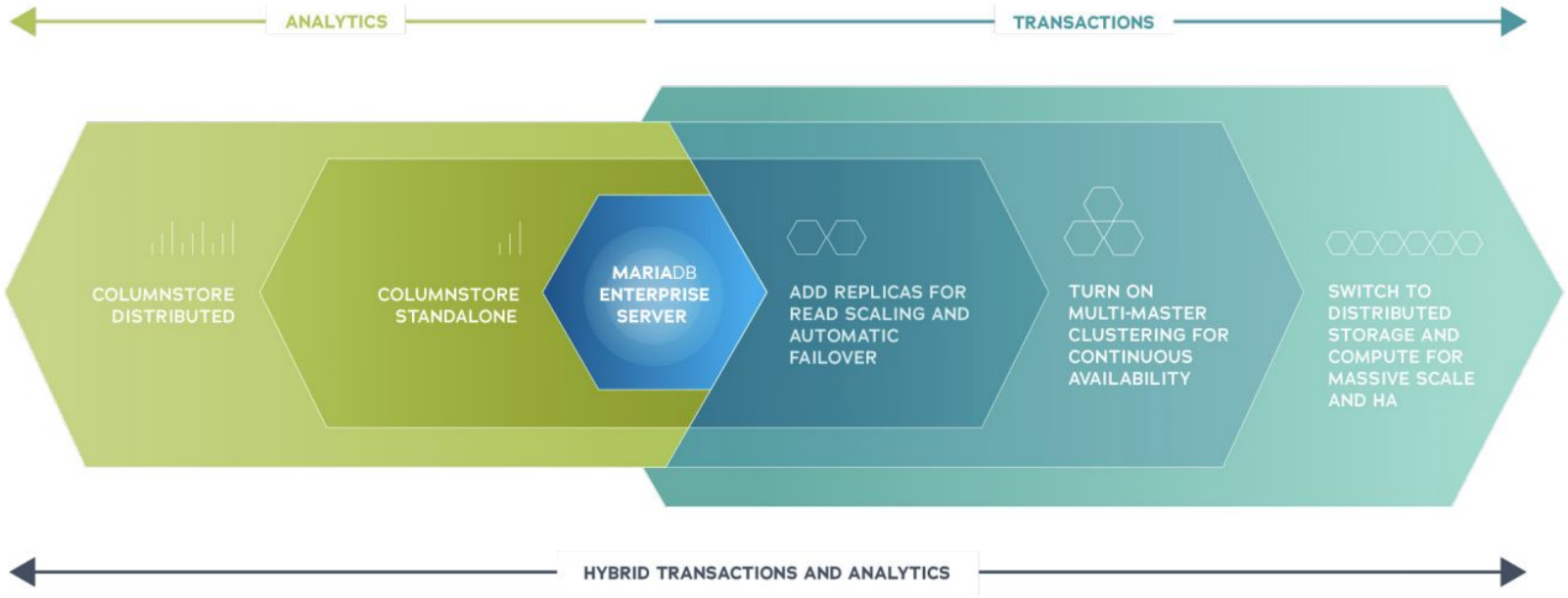
}

MariaDB

What is SkySQL

- fully-managed cloud database service
- deploys and manages databases for the customer with a few clicks
- Platform offers:
  - Enterprise Server: Single and Primary/Replica
  - Xpand - Distributed SQL: Multi-primary distributed SQL
  - Column Store
  - Serverless Analytics
  - Observability Services
  - Multicloud deployment GCP/AWS

SkySQL Products Overview



MariaDB

Customer Showcase: SAMSUNG

More than 2 Billion smart phones sold.

All SAMSUNG Android phones rely on Xpand to access the SAMSUNG Cloud.

MariaDB

Autonomous DBaaS

Create - Use

SkySQL will take care of it…

MariaDB

Autonomous DBaaS

- Disk Autoscaling
  - Based on current disk utilization
  - Prediction based on Prometheus predict_linear() over some interval (Linear Regression)
- Horizontal Scaling
  - CPU utilisation over all replicas sustained for some interval
  - Number of concurrent sessions over all replicas sustained for some interval
  - Prediction: Number of concurrent sessions is expected to hit the boundary within 4h over on the last hour data
- Vertical Scaling
  - CPU & Memory utilisation over all replicas sustained for some interval
  - Prediction on CPU & Memory utilization

Autonomous DBaaS

- SkySQL offers Terraform provider
- allows configuring any SkySQL DB topology using the Terraform`s declarative language
- https://registry.terraform.io/providers/mariadb-corporation/skysql/latest/docs

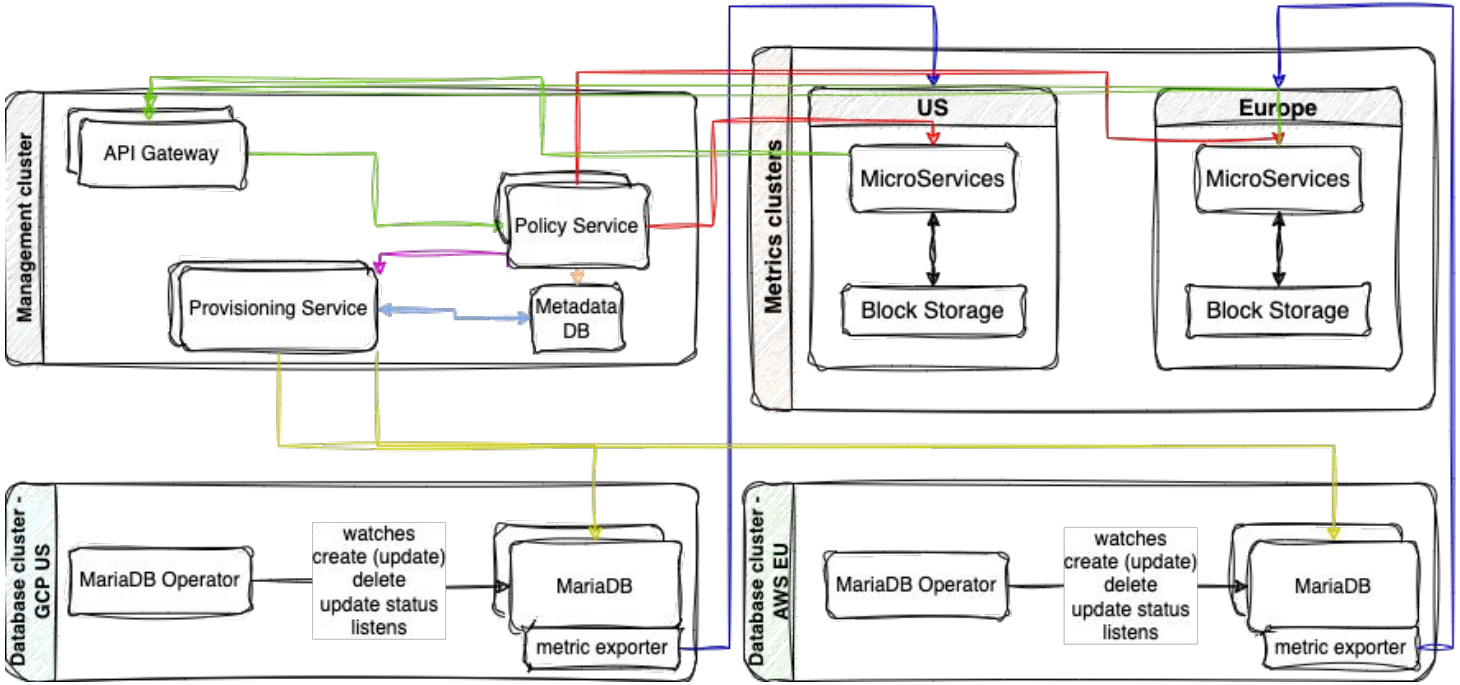
```
# Create a service
resource "skysql_service" "default" {
  service_type   = "transactional"
  topology       = "es-single"
  cloud_provider = "gcp"
  region         = "us-central1"
  name           = "myservice"
  architecture   = "amd64"
  nodes          = 1
  size           = "sky-2x8"
  storage        = 100
  ssl_enabled    = true
  version        = data.skysql_versions.default.versions[0].name
  # [Optional] Below you can find example with optional parameters how to configure a privatelink connection
  endpoint_mechanism        = "privatelink"
  endpoint_allowed_accounts = ["gcp-project-id"]
  # [/Optional]
  # The service create is an asynchronous operation.
  # if you want to wait for the service to be created set wait_for_creation to true
  wait_for_creation = true
  # You need to add your ip address in the CIRD format to allow list in order to connect to the service
  allow_list = [
    {
      "ip" : "104.28.203.45/32",
      "comment" : "office"
    }
  ]
}
```

MariaDB

MariaDB Operator

- SkySQL offers enterprise kubernetes operator
- Kubebuilder https://github.com/kubernetes-sigs/kubebuilder
- Developed on top of controller-runtime and controller-tools libraries
- *Controller-gen* generates CRDs, RBACs manifests etc, as well generate golang deepcopy methods for apimachinary interface
- Set of Controllers, which handles
  - Disk resizing, IOPs etc
  - Horizontal/Vertical scaling
  - Backups/Restore
  - Configuration Management

Autonomous Flow: multi cloud/region



MariaDB

Not the kubectl way

- Get Cluster via Cloud SDK
  - CA
  - Cluster endpoint
  - Access Token
- Connect, take care of NAT,VPC

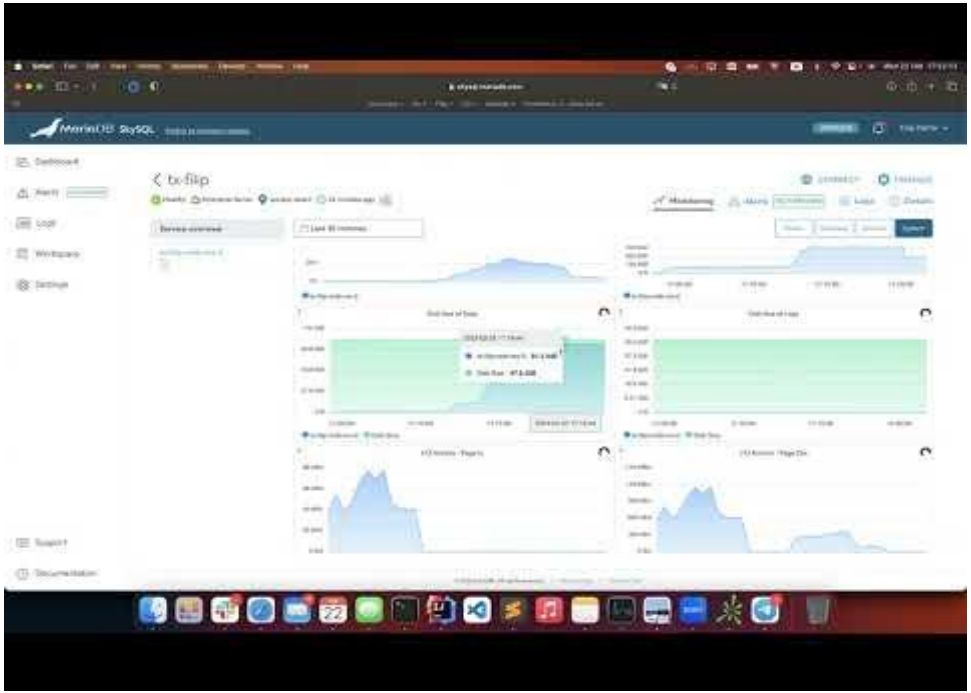
```
caCertificate, err := base64.StdEncoding.DecodeString(cluster.CaCertificate)
if err != nil {
    return errors.Wrap(err, "not able to decode cluster ca certificate")
}
tlsClientConfig := rest.TLSClientConfig{}
tlsClientConfig.CAData = caCertificate
config := &rest.Config{
    Host:            cluster.Endpoint,
    TLSClientConfig: tlsClientConfig,
    BearerToken:     cluster.AccessToken,
}

scheme := runtime.NewScheme()
if err = mdbv1alpha1.AddToScheme(scheme); err != nil {
    return errors.Wrapf(err, "could not add mariadb kubernetes to schema")
}
if err = corev1.AddToScheme(scheme); err != nil {
    return errors.Wrapf(err, "could not add corev1 kubernetes to schema")
}
// Create the client
clientSet, err := client.New(config, client.Options{
    Scheme: scheme,
})
if err != nil {
    return err
}

mdbList := &mdbv1alpha1.MariaDBList{}
if err = clientSet.List(ctx, mdbList); err != nil {
    return err
}
log.Println("Available MariaDBs for this cluster:")
for _, ns := range mdbList.Items {
    log.Printf("*\t Mariadb Name %s, Namespace %s", ns.Name, ns.Namespace)
}
```

MariaDB

Multi-Cloud Challenges

- GCP Golang sdk is structured significantly better than AWS
- GCP doesn’t allow update of existing node pool, creates new one and cordones
- AWS supports updating node pool
- GCP latency between the zones ridiculously lower, similar to single zone latency in AWS

DEMO: Disk Autoscaling



SCAN ME

Ask me Anything

MariaDB