Building a DevOps PaaS with Docker, CoreOS, Kubernetes and Apache Stratos
About Me

Lakmal Warusawithana
Vise President, Apache Stratos
Director - Cloud Architecture, WSO2 Inc
lakmal@apache.org / lakmal@wso2.com
Agenda

Presentation
- Technical overview of Apache Stratos

Demo with Apache Stratos 4.1.0-M2 developer Preview
- Setup with 3 node CoreOS cluster, Discovery service, Kubernetes master and 3 minions and flannel
- Configure Stratos
- Register Kubernetes-CoreOS host cluster to Stratos
- Deploy Docker based PHP Cartridge
- Deploy PHP application using PHP Cartridge
- Automated artifact updates
- Manual Scaling
- Autoscaling based on load average
Apache Stratos

- Apache Stratos is a highly-extensible Platform-as-a-Service (PaaS) framework that helps run Apache Tomcat, PHP, and MySQL applications and can be extended to support many more environments on all major cloud infrastructures.

- Stratos initially develop by WSO2 and last year donated to Apache Software Foundation.

- After successfully complete the incubating process Stratos now graduated as Top Level Project.
Apache Stratos L1 Architecture for VM based Cartridges

Service 1
  - Load Balancer 1
  - Service Node 1
  - Service Node 2

Service 2
  - Load Balancer
  - Service Node 1
  - Service Node 2

Foundation Services
- User Management
- Billing
- Storage

Real Time Event Bus
- jclouds

IaaS
- IaaS 1
- IaaS 2
- IaaS n

(OpenStack, vCloud, CloudStack, EC2 ... )
Apache Stratos L1 Architecture for Docker based Cartridges

- Logging
- Metering
- Artifact Distribution Coordinator
- Auto Scaler (Rule Based)
- CEP
- Cloud Controller
- Real Time Event Bus
- Message Bus
- Kubernetes Cluster 1
- Kubernetes Cluster 2
- Kubernetes Cluster n

Foundation Services
- User Management
- Billing
- Storage
Apache Stratos Cartridges

Frameworks:
- Spring
- Struts
- Joomla!

App Servers:
- Apache Tomcat 7
- JBoss
- Apache HTTP Server

Languages:
- PHP
- Java
- .NET

Databases:
- MySQL
- PostgreSQL
- Cassandra
- MongoDB

Legacy Apps:
True Flexibility for Custom Cartridges

- In most cases, you have to make-do with what’s available and work your solution around it
- With the Apache Stratos cartridge model, you can create a custom service without having to deal with any limitations or boundaries.
  - Cartridge can be fully configured (installed all software, configuration, etc.) or
  - zero configured (enable cartridge user to install and configure what they want) or
  - something in-between
- This will allow you to customize your PaaS to be in sync with your current business workflows
Multi-factored Auto Scaling

What is it?

- Scaling algorithm can use multiple factors. such as
  - Load average of the instance
  - Memory consumption of the instance
  - In-flight request count in LB
Multi-factored Auto Scaling...

- Capable of predicting future load
  - Real time analysis of current load status using CEP integration
  - Predict immediate future load based on CEP resulting streams
  - Predicting equation $s=ut + \frac{1}{2}at^2$
  - $s$=predicted load, $u$=first derivative of current average load, $t$= time interval , $a$=second derivative of current load

Why should one care?
- Maximise resource utilization
- Easy to do capacity planning
- Dynamic load based resource provisioning
- Optimizing across multiple clouds
Scalable and Dynamic Load Balancing

How Scalable it is?
- In theory infinite
  - horizontal scaling
  - limited by resource (instance capacity) availability

How Dynamic it is?
- Load Balancers are spawned dynamically
  - LB too is a cartridge
- In case of multi-cloud, multi-region, LB can scale per cloud/region
- Per service cluster LB
What is unique about Stratos

- Cartridge based LB model
- Can bring any third-party LB
  - HAProxy, nginx, AWS ELB
  - As easy as plugging into LB extension API
Smart Policies

What are the smart policies?
- Auto scaling
- Deployment

Auto scaling policy
- Define thresholds values pertaining scale up/down decision
- Auto Scaler refer this policy
- Defined by DevOps

Deployment policy
- Defined how and where to spawn cartridge instances
- Defined min and max instances in a selected service cluster
- Defined by DevOps based on deployment patterns
Smart Policies

Why should one care?
- Can provide cloud SLA

What are the advantages?
- Make DevOps life easy
  - help keep to SLA
- Make SaaS app delivery life easy
  - do not have to worry about availability in application layer
Multi-tenancy

What MT model does it support?
- Container MT
  - virtual Machine, LXC, Docker
- In-container MT
  - within VM/LXC/Docker tenancy

What is unique?
- Can have high tenant density

What are the advantage of this model?
- Optimizing resource utilization
  - by sharing resource such as CPU, memory across tenants
  - low footprint, based on utilization/usage of the tenants app
- No need dedicated resource allocation for tenants
Cloud Bursting

What is it?
○ Expanding/provisioning application into another cloud to handle peak load.

Why Should one care?
○ Resource peak time can be off-loaded to third party clouds/resources

What is unique about it?
○ Can off-load to any cloud
   - Private, Public and Hybrid
○ Easy to managed with the model of LB per busting cloud
What are the advantages?
- Make DevOps life easy
- Low TCO, and higher utilization existing dedicated resources
Logging, Metering and Monitoring

What details are?
- Instance up/down time
- Each and every instances health status
  - application health, load average, memory consumption
- Application logs

Why should one care?
- Centralize view for all logging, metering and monitoring

What are the advantages?
- DevOps life easy
  - centralize log viewer
  - centralize dashboard
- Easy to throttling
Apache Stratos next release is mainly into
- Docker based cartridge support
- integration with CoreOS
- integration with Kubernetes
- integration with flannel
- integration with discovery service and build in docker registry support

Support docker top of VM
- provide two level of scalability
- support for integrated with any existing IaaS
Stratos Architecture with Docker Support
Demo

Using Apache Stratos 4.1.0-m2 developer preview

- Setup with 3 node CoreOS cluster, Discovery service, Kubernetes master and 3 minions and flannel
- Configure Stratos
- Register Kubernetes-CoreOS host cluster to Stratos
- Deploy Docker based PHP Cartridge
- Deploy PHP application using PHP Cartridge
- Automated artifact updates
- Manual Scaling
- Autoscaling based on load average
More Information!

- Apache Stratos

- Apache Stratos 4.1.0-m2 developer preview
  [https://cwiki.apache.org/confluence/display/STRATOS/4.1.0+Stratos+M2+Developer+Preview](https://cwiki.apache.org/confluence/display/STRATOS/4.1.0+Stratos+M2+Developer+Preview)

- Why Apache Stratos is the preferred choice in the PaaS Space

- Sneak peak into Apache Stratos 4.0.0