Optum Global Solutions

OpenShift Enterprise V3.1

Training Presentation for Open shift Enterprise

Sreejith Kaimal

Updated for OSE Prod
GA. April 2016

January 15,, 2016

IaaS, PaaS and SaaS

IaaS Infrastructure As A Service

- Provider spins up virtual devices for you on demand. (e.g. NGIS via ESC)
- User provides the amount of RAM, CPU and Storage – Provisioned.
- Templates provide OS platform. E.g.. Linux or windows.
- Mostly provisioned as Virtual Machines.

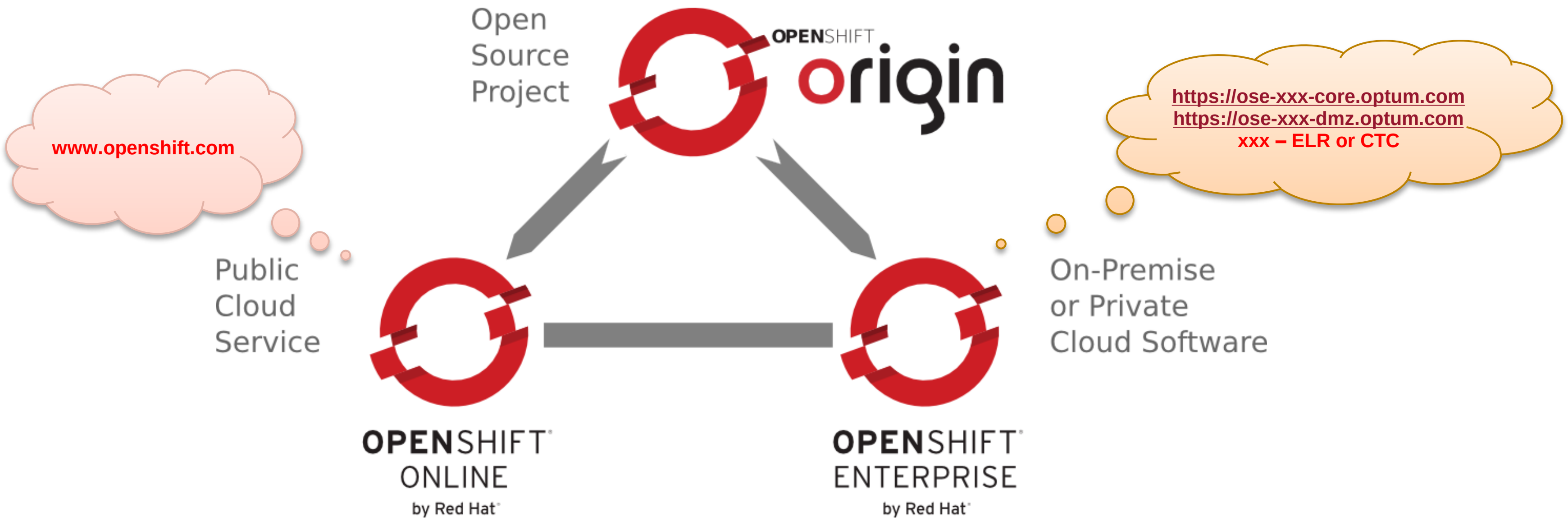
SaaS Software As A Service

- Software ready to use, available over existing shared multi tenant Infra.
- Limited to what service provider has to offer.
- Targeted at non technical users – e.g., Gmail, Salesforce, QuickBooks etc.
- Data is stored alongside other clients.

PaaS Platform As A Service

- Combines IaaS and SaaS in a variety of different ways.
- Can get (scalable) infrastructure setup via self service provisioning
- Software can be installed on the infrastructure using available templates.
- Minimal time to setup a development environment.

Three Versions of Open Shift



www.openshift.com
Open Source Project
OPENSHIFT origin
https://ose-xxx-core.optum.com
https://ose-xxx-dmz.optum.com
xxx – ELR or CTC
Public Cloud Service
On-Premise or Private Cloud Software
OPENSHIFT ONLINE by Red Hat
OPENSHIFT ENTERPRISE by Red Hat

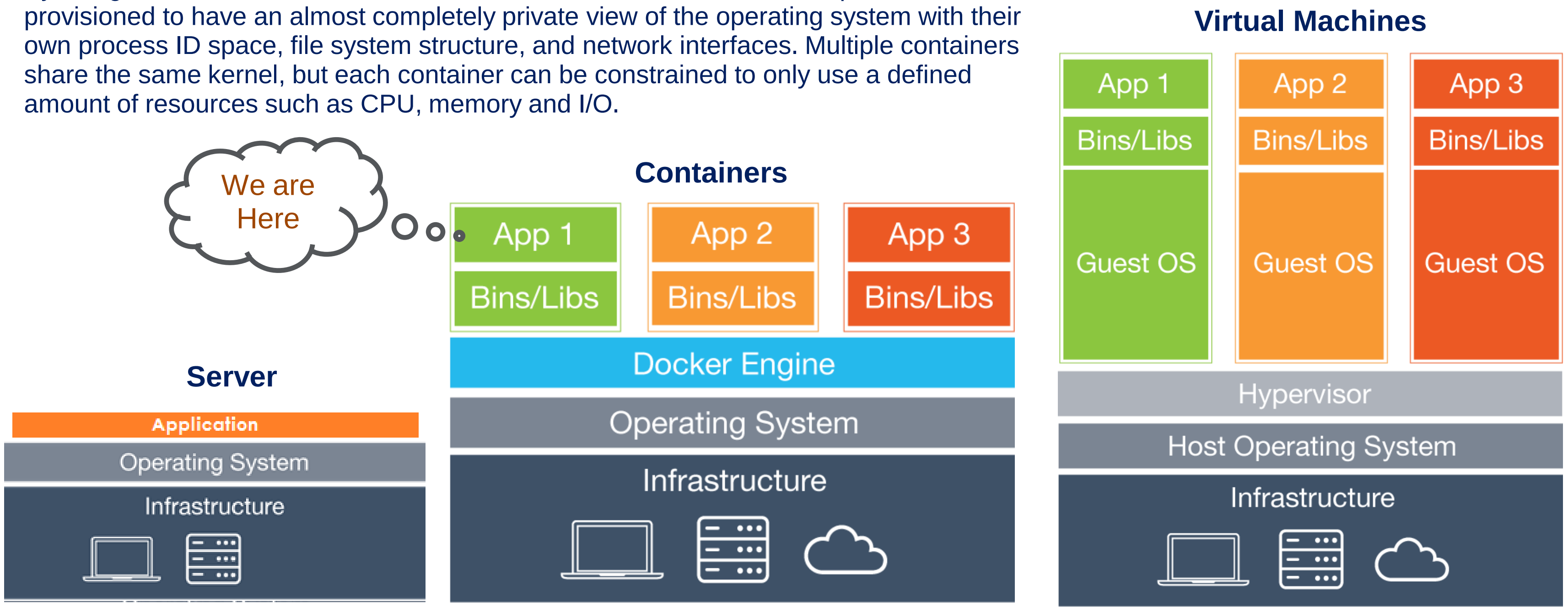
- **OpenShift Origin** : The free & open source version of OpenShift
- **OpenShift Online** : Hosted on Red Hat, its available over the internet with Free & Paid plans to build & deploy personal Applications.
- **OpenShift Enterprise** :Enterprise cloud Implementation on In-House data center, gives full features of OpenShift with Sys-Admins to have control on the Infrastructure.

What is in OpenShift V3..

Introduction to Containers

Docker containers wrap up a piece of software in a complete file system that contains everything it needs to run: code, runtime, system tools, system libraries – anything you can install on a server. This guarantees that it will always run the same, regardless of the environment it is running in.

By using containers, resources can be isolated, services restricted, and processes provisioned to have an almost completely private view of the operating system with their own process ID space, file system structure, and network interfaces. Multiple containers share the same kernel, but each container can be constrained to only use a defined amount of resources such as CPU, memory and I/O.



- Stand up new physical servers
- OS & Hardware are pre-decided.
- Takes hardware , network & user provisioning.
- No scale in
- Vendor lock in + PLM issues

- Contains only the Application specific Bin/Lib. (lightweight)
- Share OS level resources with others.
- Eliminate environment inconsistency.
- Can be installed on any infrastructure.
- Share with team 'Ship it as container'
- Used shared OS
- Less CPU, RAM , Storage - Spin Faster than VM
- More containers per machine

- Includes everything required in a standalone OS
- Copy the entire library, software installation and content.
- Updates and patches applied to individual VM
- Individual maintenance.
- Use hardware virtualization. Hence limits number of VM on each box.

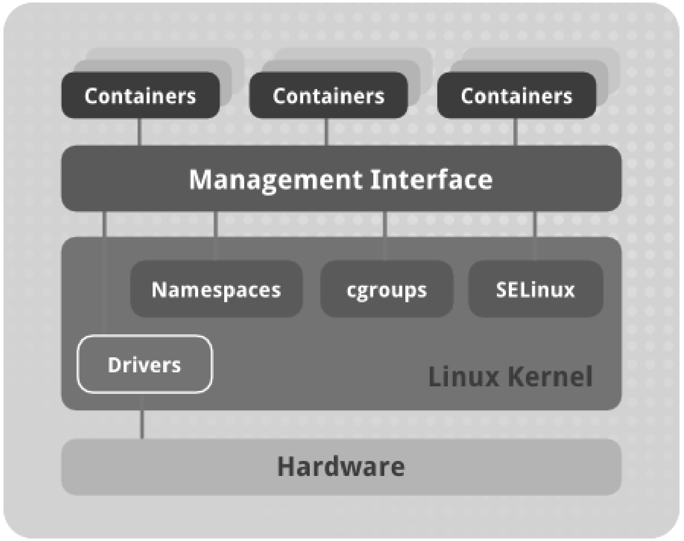
Linux Containers ..

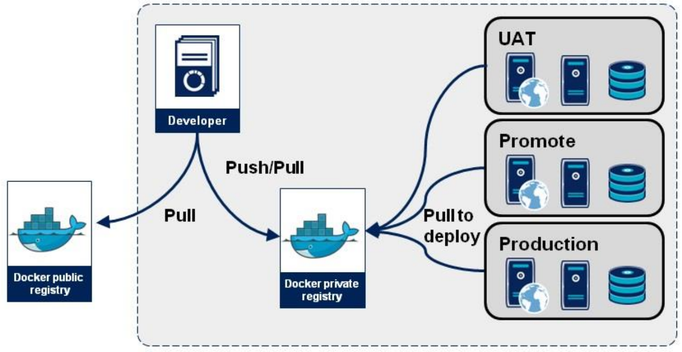
Run Anything, Anywhere ...

Red Hat Enterprise Linux 7 implements Linux Containers using core technologies such as Control Groups (Cgroups) for Resource Management, Namespaces for Process Isolation, SELinux for Security, enabling secure multi-tenancy and reducing the potential for security exploits.

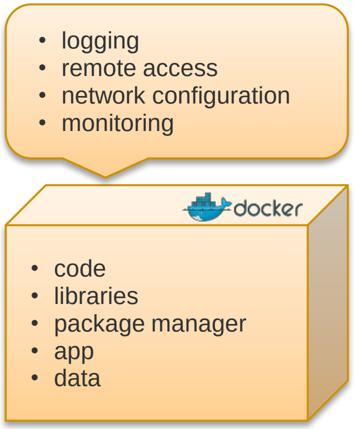
- **Cgroups** ensure that a single container cannot exhaust a large amount of system resources, thus preventing some denial-of-service attacks.
- Security-Enhanced Linux (**SELinux**) is an implementation of a mandatory access control (MAC) mechanism, multi-level security (MLS), and multi-category security (MCS) in the Linux kernel.
- **Namespaces** viz, ***Mount***(isolate the set of file system mount points),***UTS*** (allows each container to have its own hostname and NIS domain name),***IPC*** (interprocess communication),***PID*** (allow processes in different containers to have the same PID),***Network*** (provide isolation of network controllers, system resources associated with networking, firewall and routing tables)

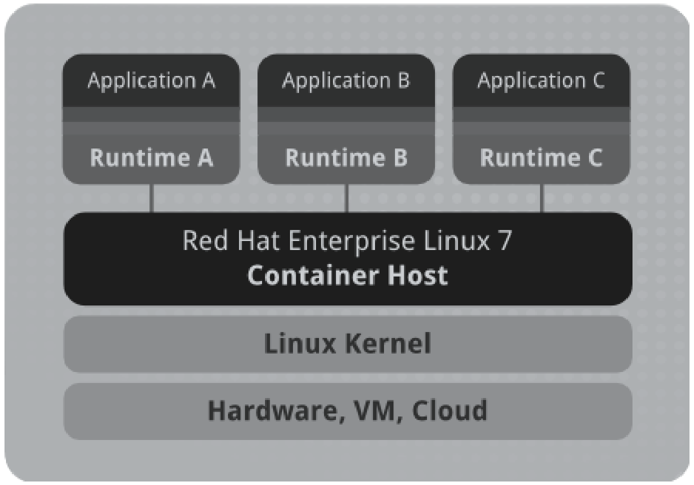
Containers are lightweight
- Typical laptop runs 10-100 containers easily
- Typical server can run 100-1000 containers





Docker Image Lifecycle





FYI Only : Not required for cloud development teams..

OpenShift V3 – Familiarize with Terminologies.

DOCKER	• Platform for developing , shipping & running applications using container virtualization technology. • A container is a stripped-to-basics version of a Linux operating system. An *image* is software you load into a *container*. • Consists of multiple tools Docker Engine, Docker Hub, Docker Machine, Docker Swarm, Kinematic etc.. OSE uses Kubernetes to manage the containers.
DOCKER ENGINE	• Software that enables containers to be built, shipped and Run. (Docker daemon) • Docker Engine uses Linux Kernel namespaces and Control Groups. • Namespaces give isolated workspaces.(pid, net, ipc, mnt,uts etc..)
DOCKER CLIENT & DAEMON	• Client / Server Architecture for connecting to Docker engine • Client takes user input & sends them to daemon. Daemon builds , runs and distributes. • Client and daemon can run on same or different hosts. • CLI client and GUI Client (Kitematic)
KUBERNETES	• Google project that OSE V3 uses to manage containers. • Provides mechanism to deploy, maintain and scale applications. • Uses Master & Replication controller to orchestrate. • Has components to load balance. • Kubelet : An agent type component that watches over the Nodes.

OpenShift V3 Familiarize with Terminologies

CONTAINER	• lightweight mechanisms for isolating running processes so that they are limited to interacting with only their designated resources • Typically, each container provides a single service (often called a "micro-service") • It is software bundled in a complete file system that contains everything it needs to run e.g., Code, runtime, system tools, system libraries
DOCKER IMAGE	• A packaging technology. • A Docker image is a binary that includes all of the requirements for running a single Docker container, as well as metadata describing its needs and capabilities • Each different image is referred to uniquely by its hash , shortened to 12 bytes. • Your application would be bundled as an image which would be deployed on the OSE v3
DOCKER REGISTRY	• A Docker registry is a service for storing and retrieving Docker images. • A registry contains a collection of one or more Docker image repositories. • Each image repository contains one or more tagged images. • OpenShift can also supply its own internal registry for managing custom Docker images
POD	• A Pod holds one or more Docker containers together. • Kubernetes Object that groups related Docker containers that share network, file system or memory together. • Scaling is achieved by running multiple instances of a Pod

OpenShift V3 Stack

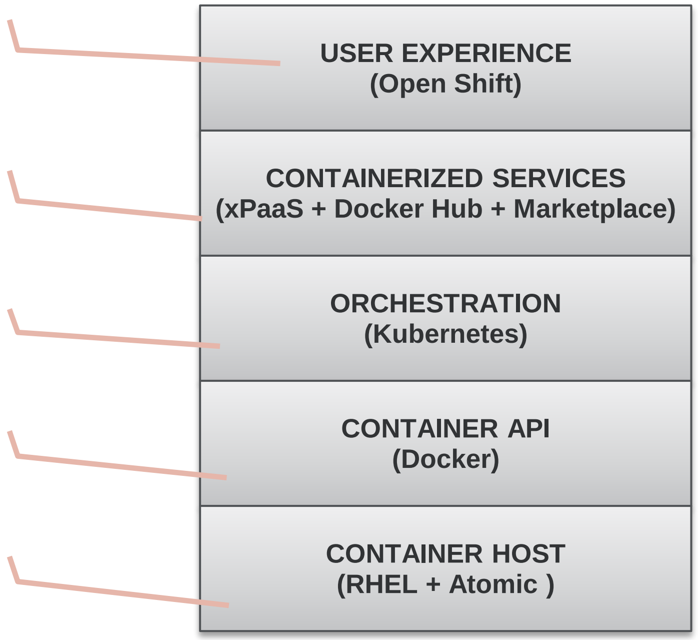
Developer & Administrative experience for Management. (CLI, JBDS, Web Console)

1. xPaaS – Application Server(JBoss) or Integration etc.
2. Docker Hub – Containers & Images
3. Marketplace – where custom container images are available

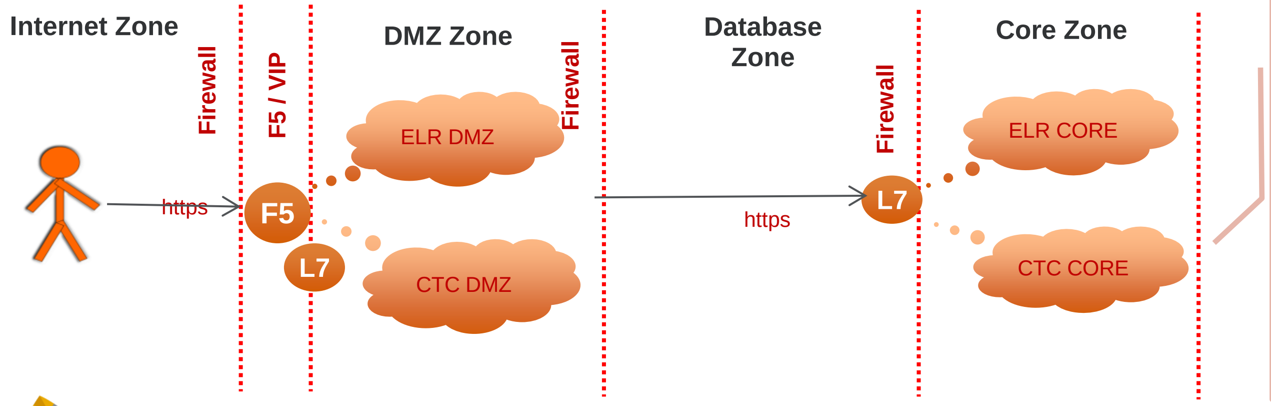
Kubernetes – Manages containers on multiple hosts, scaling & orchestration.

Docker Engine – Docker services running on OS to enable containerization

Host – The RHEL 7 OS Atomic version optimized for Docker.



USER EXPERIENCE (Open Shift)
CONTAINERIZED SERVICES (xPaaS + Docker Hub + Marketplace)
ORCHESTRATION (Kubernetes)
CONTAINER API (Docker)
CONTAINER HOST (RHEL + Atomic)

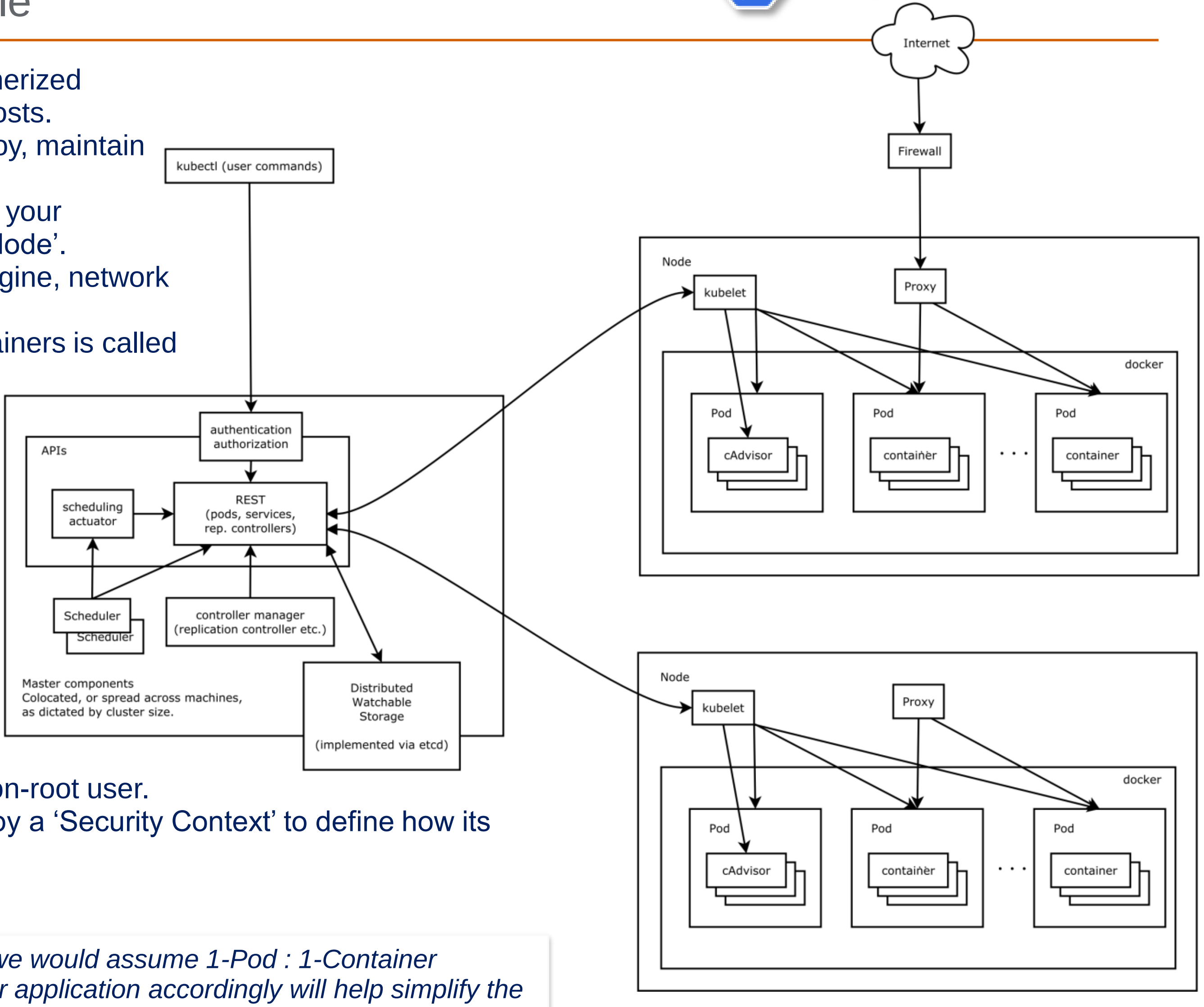


OSE V3 has presence in both Core & DMZ zones, both CTC & ELR DC, making it easier to position apps / services as per the security / load requirements.
This can help match the application with the purpose with which it is built. Reduces the need for additional FW rules / PEX etc.
Each instance of OSE is treated as a separate PaaS environment. Each environment provisioning is handled individually via ESC.

{koo-ber-nay'-tace}

Kubernetes by Google

1. System for managing containerized application across multiple hosts.
2. Provides mechanism to deploy, maintain and scale applications.
3. A physical machine on which your containers are running is a 'Node'.
4. Each node runs a Docker Engine, network proxy and load balancer.
5. A group of one or more containers is called a 'Pod'
6. 'Labels' are key value pairs that go with each Object in Kubernetes environment, can be used to search pods.
7. 'Replication Controllers' handle the scaling up and down.
8. 'Services' provide an abstract way to reach a set of labelled 'Pods'



kubernetes
Internet
Firewall
kubectl (user commands)
authentication authorization
APIs
scheduling actuator
REST (pods, services, rep. controllers)
Scheduler
Scheduler
controller manager (replication controller etc.)
Master components Colocated, or spread across machines, as dictated by cluster size.
Distributed Watchable Storage (implemented via etcd)
Node
kubelet
Proxy
docker
Pod
cAdvisor
Pod
container
Pod
container
Node
kubelet
Proxy
docker
Pod
cAdvisor
Pod
container
Pod
container

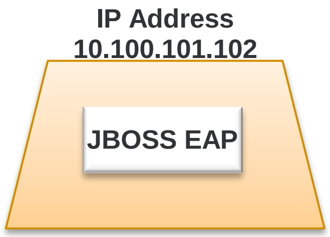
6. All containers run a single non-root user.
7. Each container is governed by a 'Security Context' to define how its secured.
8. Uses CLI for administration.

For easy resource management, we would assume 1-Pod : 1-Container deployment model. Designing your application accordingly will help simplify the scaling model.

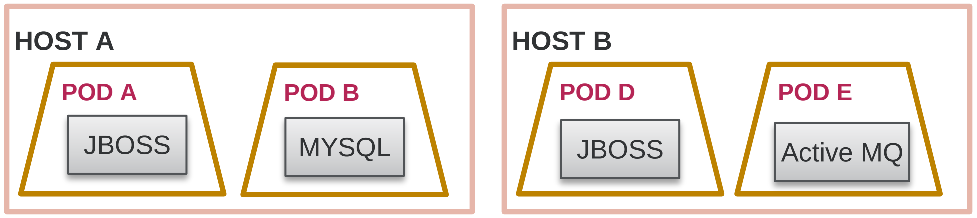
OpenShift V3..

MASTER	• Host or hosts that contain Master components. • API Server validates & configures Pods, synchronizes • ETCD stores state of master from which other components watch & sync. E.g Replication Controller. • Replication controller watches ETCD and forces API Server to sync other.
NODES	• OpenShift creates Nodes from Physical or Virtual systems. • Nodes provide runtime environment for Containers • Each node will have a Docker Installation, a Kubelet and a Service Proxy • Kubelet runs on each node, does a 20 second sync up & self check.
PROJECT	• A project is a Kubernetes namespace with additional annotations, for user resource management. • A project allows a community of users to organize and manage their content. • Each project will have its own Objects (pods, services etc.) Rules , Constraints and Service accounts.
BUILDS	• Process of transforming source code into runnable image. • OpenShift provides a build system to perform a Docker Build, S2I (source to image) & Custom build. • An image stream allows build or deployment to OSE.

POD

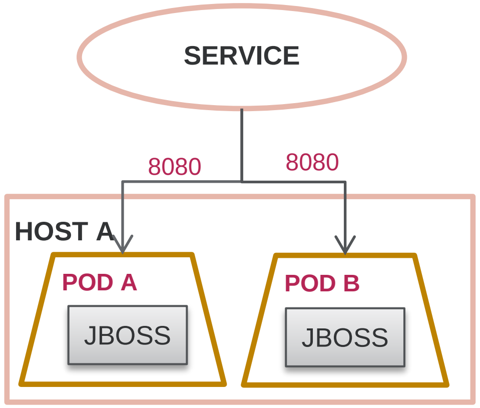


1. OpenShift leverages the Kubernetes concept of a pod.
2. Pods are the rough equivalent of OpenShift v2 gears, with containers the rough equivalent of v2
3. cartridge instances.
4. Each Pod has a single IP address. All containers in the Pod share the same IP and Port space.
5. Containers within Pod should share ports – conflicts will not be allowed.
6. Each host can have one or more PODs on it.
7. A Service is a method to group & load balance. When a service is used, its accessed via IP:port . Services are not external facing IP.

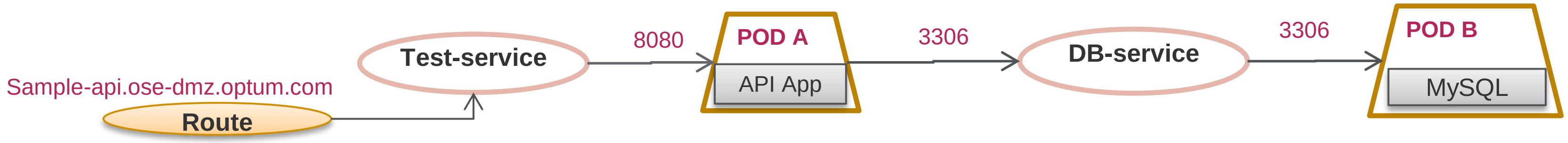


1. Pods can be "tagged" with one or more **labels**
2. Each host can have one or more Pods
3. Each Pod can have one or more container images

Ps: We would assume that each pod will have only 1 image.



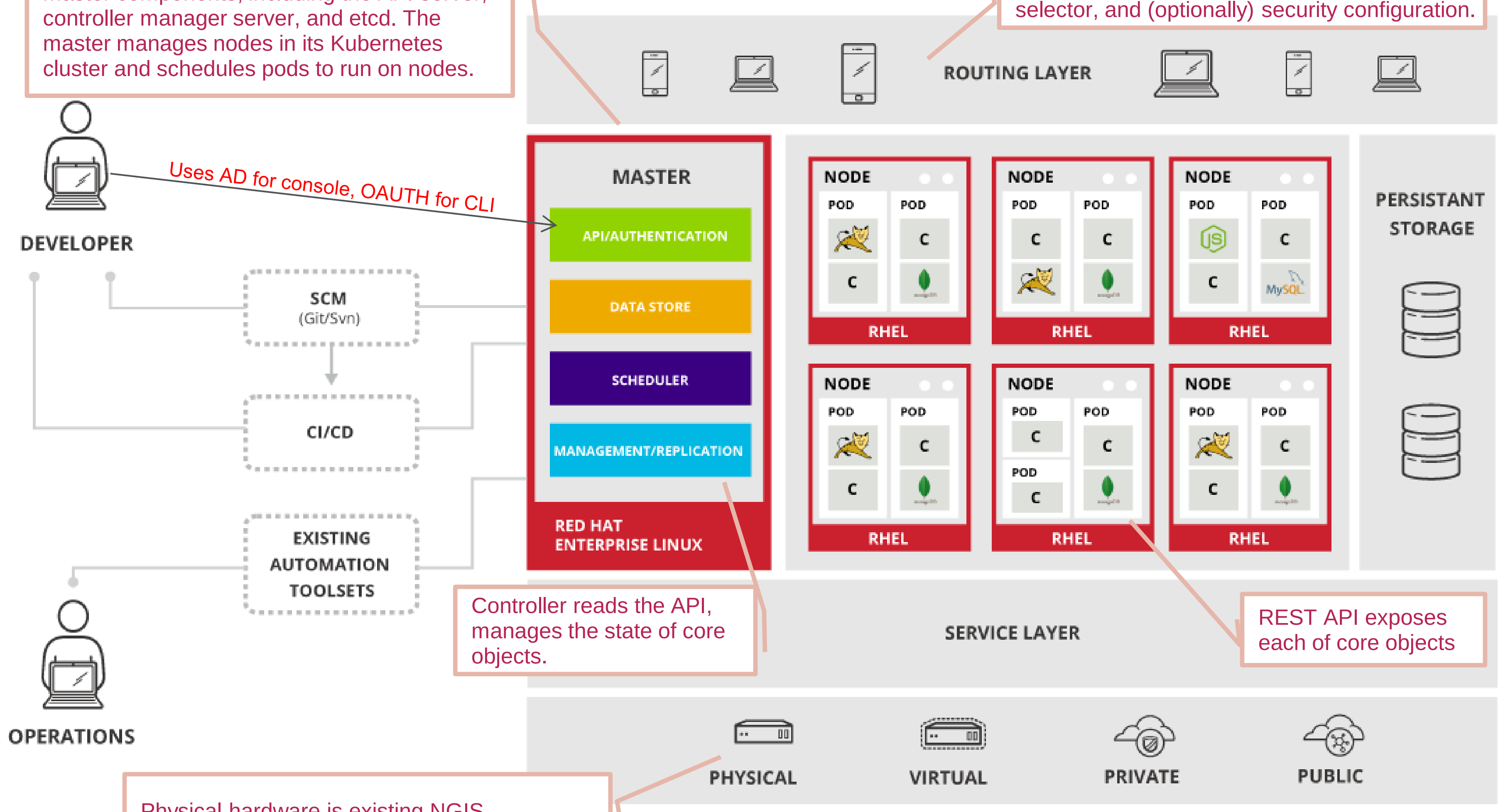
1. Pods can be "tagged" with one or more labels.
2. The labels are stored in key/value format in the metadata hash.
3. A Service is a method to group & load balance. When a service is used, its accessed via IP:port . Services are not external facing IP.
4. A service uses a label selector to find all the containers running that provide a certain network service on a certain port.
5. Service gets a virtual IP.
6. A replication controller ensures that a specified number of replicas of a pod are running at all times.
   - If pods exit or are deleted, the replica controller acts to instantiate more up to the desired number.
   - If there are more running than desired, it deletes as many as necessary to match the number.
7. An OpenShift route is a way to expose a service by giving it an externally-reachable hostname



OpenShift V2 and V3 Comparison.

OpenShift V2	OpenShift V3
PaaS with limited supported cartridges, libraries & system tools. (RHEL V6)	Micro services-based architecture of smaller, decoupled units that work together. (RHEL V7 Atomic) - Docker
Each Application can comprise of one web framework. Each cartridge can do only one type of work. Cartridges share the libraries that are installed on the OS.	Components/App runs in Docker containers. Containers have a standalone dependency stack that does not interfere with other containers. Scheduling & Management provided by Kubernetes.
Gears contain one or more cartridge, Gear scales up 1-N on http traffic.	A pod will have one or more container images. 1 Pod 1 Container , scales up by replication of pods.
Scaling is done by cloning the existing gear. Admins/Dev can change configuration	OSE containers are 'immutable' code + libraries + configuration & data is maintained as a snapshot.
HA Proxy triggers the scaling process.	Scaling managed by Kubernetes using Pods + Services.
Administration via Broker	Administration is done via REST-Api's + kubernetes.
Requires at least one web framework in Git repo.	Choice of 'image' source is configurable, image can be located outside OpenShift itself.
Build and deploys were done using resources from existing gears. App would go down while deploying.	Builds happen in different containers. Build results are available in 'internal registry' to be launched or rolled back.
Dependencies were to be declared up-front	Can specify runtime dependencies.
Node based routing.	Platform Based Routing

OpenShift V3 Architecture



The master is the host or hosts that contain the master components, including the API server, controller manager server, and etcd. The master manages nodes in its Kubernetes cluster and schedules pods to run on nodes.

Routing layer allows exposing a 'service' Each route consists of a route name, service selector, and (optionally) security configuration.

Controller reads the API, manages the state of core objects.

REST API exposes each of core objects

Physical hardware is existing NGIS infrastructure.

https://access.redhat.com/documentation/en/openshift-enterprise/version-3.0/openshift-enterprise-30-architecture/

Requesting Access

The first step in getting started with OpenShift is to request access through the Enterprise Service Catalog. https://servicecatalog.uhc.com

1. Go to Infrastructure -> Platform Services Find Openshift on the right pane.
2. Fill the self service form
3. GL would be used for Billing Purposes. (use the billing app GL)
4. Use an application ID if this is for a system account. (use secure for creating system id prior to this step.)
5. Select the Datacenter, Region, ENV and Zone. This selection must be in accordance with the SLD / application specific requirement. If you are unsure – please reach out to your Systems Engineer / Architects.
6. Select project sizing (Small-5XL) *easy way is to select is to ask close to 2.5 times the heap space that you would need on your server. E.g, for 512Mi heap space then ask for 2Gi Small. For AEM type containers, follow the AEM templates sizing. Do not request for more than what you really need.*

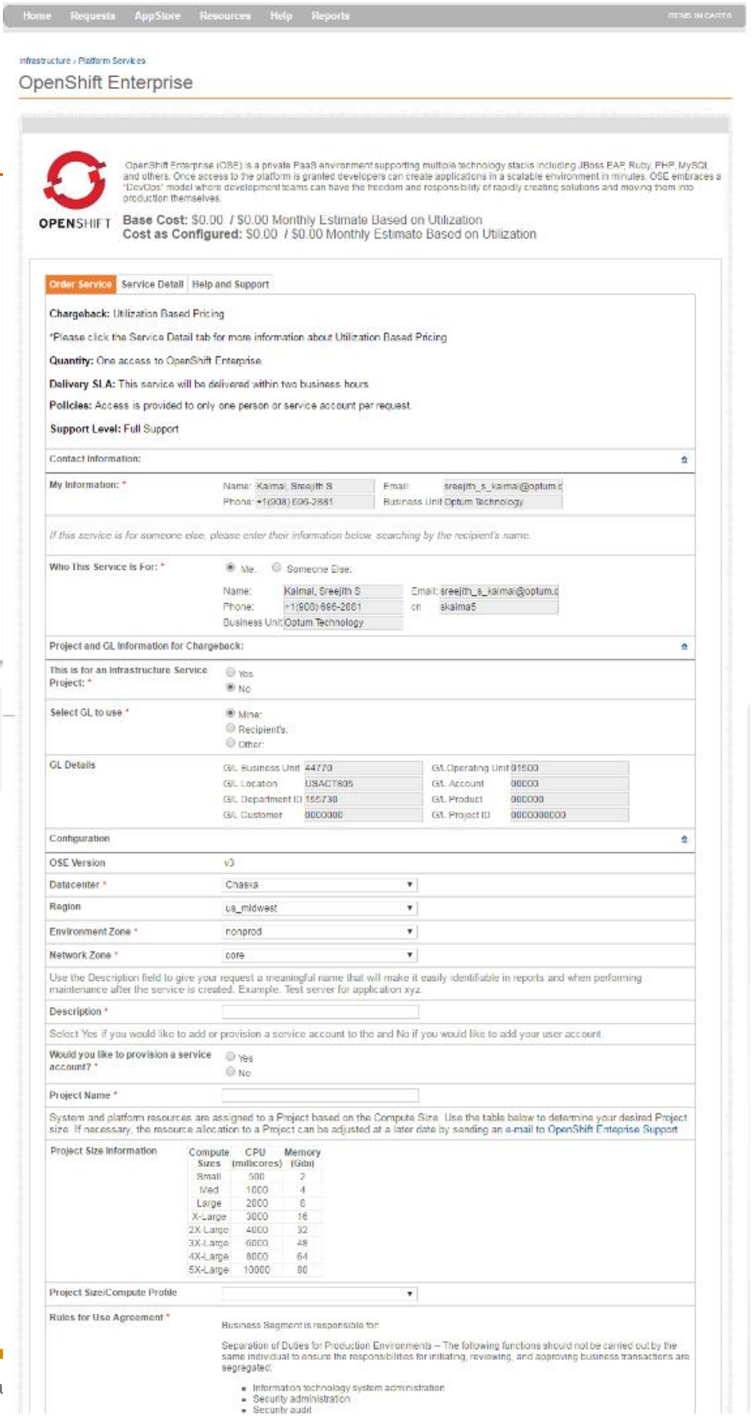
OSE Core:
Chaska - https://ose-ctc-core.optum.com
Elk River - https://ose-elr-core.optum.com

OSE DMZ:
Chaska - https://ose-ctc-dmz.optum.com
Elk River - https://ose-elr-dmz.optum.com

Compute Sizes	CPU (millicores)	Memory (Gibi)	PVC	Pods	Services	Replication Controllers	Resource Quota	Secrets
Small	500	2	2	4	2	6	1	20
Med	1000	4	4	8	4	12	1	20
Large	2000	8	8	16	8	24	1	40
X-large	3000	16	16	32	16	48	10	40
2X-Large	4000	32	24	64	32	86	10	60
3X-Large	6000	48	32	96	48	144	10	60
4X-Large	8000	64	40	128	64	192	10	80
5X-Large	10000	80	80	160	80	240	10	80

https://servicecatalog.optum.com/sc/catalog.product.aspx?parent_category_id=_X_&category_id=Platform%20Services&product_id=OpenShiftEnterprise

OPTUM™



Infrastructure › Platform Services

OpenShift Enterprise

OpenShift Enterprise (OSE) is a private PaaS environment supporting multiple technology stacks including JBoss EAP, Ruby, PHP, MySQL and others. Once access to the platform is granted developers can create applications in a scalable environment in minutes. OSE embraces a "DevOps" model where development teams can have the freedom and responsibility of rapidly creating solutions and moving them into production themselves.

Base Cost: $0.00 / $0.00 Monthly Estimate Based on Utilization
Cost as Configured: $0.00 / $0.00 Monthly Estimate Based on Utilization

Chargeback: Utilization Based Pricing

*Please click the Service Detail tab for more information about Utilization Based Pricing

Quantity: One access to OpenShift Enterprise

Delivery SLA: This service will be delivered within two business hours

Policies: Access is provided to only one person or service account per request.

Support Level: Full Support

Contact Information:

My Information: * Name: Kaimal, Sreejith S Email: sreejith_s_kaimal@optum.c
Phone: +1(908) 696-2881 Business Unit Optum Technology

If this service is for someone else, please enter their information below, searching by the recipient's name.

Who This Service Is For: * [✓ Me.] Someone Else:
Name: Kaimal, Sreejith S Email: sreejith_s_kaimal@optum.c
Phone: +1(908) 696-2881 cn skaima5
Business Unit Optum Technology

Project and GL Information for Chargeback:

This is for an Infrastructure Service Project: * Yes [✓ No]

Select GL to use * [✓ Mine:] Recipient's: Other:

GL Details

G/L Business Unit	44770	G/L Operating Unit	01500
G/L Location	USACT805	G/L Account	00000
G/L Department ID	155730	G/L Product	000000
G/L Customer	0000000	G/L Project ID	0000000000

Configuration

OSE Version v3
Datacenter * Chaska
Region us_midwest
Environment Zone * nonprod
Network Zone * core

Use the Description field to give your request a meaningful name that will make it easily identifiable in reports and when performing maintenance after the service is created. Example. Test server for application xyz.

Description *

Select Yes if you would like to add or provision a service account to the and No if you would like to add your user account.

Would you like to provision a service account? * Yes No

Project Name *

System and platform resources are assigned to a Project based on the Compute Size. Use the table below to determine your desired Project size. If necessary, the resource allocation to a Project can be adjusted at a later date by sending an e-mail to OpenShift Enteprise Support

Project Size Information

Compute Sizes	CPU (millicores)	Memory (Gibi)
Small	500	2
Med	1000	4
Large	2000	8
X-Large	3000	16
2X-Large	4000	32
3X-Large	6000	48
4X-Large	8000	64
5X-Large	10000	80

Project Size/Compute Profile

Rules for Use Agreement *

Business Segment is responsible for:

Separation of Duties for Production Environments – The following functions should not be carried out by the same individual to ensure the responsibilities for initiating, reviewing, and approving business transactions are segregated:

- Information technology system administration
- Security administration
- Security audit

Terminology

APPLICATION	• Application means one or more components. • For example a container may have only presentation layer & depend on another container for services or data. This allows them to individually scale. • This way components can be shared.
IMAGE	• Equivalent of cartridge on V2 • A Docker image is a binary that includes all of the requirements for running a single Docker container, as well as metadata describing its needs and capabilities. • Images are stored in Docker registry. • Single image, multiple versions allow versioned development.
PROJECT	• Equivalent of 'Domain' in V2. • It's a Kubernetes namespace with additional annotation . • Can have separate name, display name and description. • Each project scopes its own set of objects, policies , constraints & Service accounts.
MASTER	• Equivalent to 'Broker' in V2. • The host that contains master components like API Server, etcd, controller manager etc. • Performs health check on Nodes. • The master manages nodes in its Kubernetes cluster and schedules pods to run on nodes

Optum Developer Links

Use Optum Developer as source of truth for accurate details on training tools, development methodologies and support stack. Below are the links for Optum Developer.

https://www.optumdeveloper.com/content/odv-optumdev/optum-developer/en/getting-started/paas/getting-started-with-openshift-3.html

https://www.optumdeveloper.com/content/odv-optumdev/optum-developer/en/developer-centers/paas-developer-center/devops-java--openshift-enterprise-/openshift-v3-technical-resources/whats-new-in-openshift-v3.html

https://www.optumdeveloper.com/content/odv-optumdev/optum-developer/en/developer-centers/paas-developer-center.html

Have Questions or need support on General topics ? Visit the one connect community, post your question here.

https://oneconnect.uhg.com/groups/openshift-enterprise-users-group

Optum Global Solutions

Thank you.

Sreejith Kaimal
Sr.Architect
Provider Portal Applications