Red Hat Reference Architecture Series

OpenShift Enterprise 3 Architecture Guide - planning, deployment and operation of an Open Source Platform as a Service

Christoph Görn

Version 1.0.1, 2015-09-24

Table of Contents

Comments and Feedback

In the spirit of open source, we invite anyone to provide feedback and comments on any of the reference architectures. Although we review our papers internally, sometimes issues or typographical errors are encountered. Feedback allows us to not only improve the quality of the papers we produce, but allows the reader to provide their thoughts on potential improvements and topic expansion to the papers. Feedback on the papers can be provided by emailing refarch-feedback@redhat.com. Please refer to the title within the email.

Staying In Touch

Join us on some of the popular social media sites where we keep our audience informed on new reference architectures as well as offer related information on things we find interesting.

Like us on Facebook

https://www.facebook.com/rhrefarch

Follow us on Twitter

https://twitter.com/RedHatRefArch

Plus us on Google+

https://plus.google.com/u/0/b/114152126783830728030/

Executive Summary

OpenShift 3 is built around a core of application containers powered by Docker, with orchestration and management provided by Google's Kubernetes, on a foundation of Red Hat Enterprise Linux Server. OpenShift Enterprise is Red Hat's enterprise level product based on OpenShift 3 Origin (community upstream).

This infrastructure architecture illustrated by the ACME Corp example shows the infrastructure agnostic nature of OpenShift Enterprise 3: given a network and host system infrastructure, we can utilize it to provide a platform for developers to run their business applications.

Business Problems addressed by OpenShift Enterprise 3

Within this Architecture Guide we will address two major business problems:

- **deployments shall be fast** - regardless of the deployment trigger, may it be a security patch or an update to online shop features or base operating system components
- **developers shall reuse common components** - common components shall be deployed onto a platform and shall be reused and consumes as a service, as long as it is not proven they cannot be reused

OpenShift Enterprise 3 will enable businesses to deploy applications in a much faster and highly automated process. This is not only due to the fact that Continuous Testing and Integration (CT/CI) is an integral part of the OpenShift Enterprise 3 Platform, but also due to the fact that a highly integrated developer experience is delivered with OpenShift Enterprise 3.

A high degree of automation (from development, through CT/CI, to production) enables developers and operators to react faster on changing business demands.

Foreword

Within this Architecture Guide the planning, deployment and operation of a OpenShift Platform as a Service (PaaS) [1: https://www.redhat.com/en/technologies/cloud-computing/openshift] for the ACME Corporation is described. Throughout this Architecture Guide a fictional company, ACME Corporation, will be used. This Architecture Guide will describe the organization, business aspects, roles and tools of ACME Corp. so that you can easily understand the requirements.

For the purpose of this document, a Architecture Guide aims to target Solution Architects, Infrastructure Architects and Lead System Administrators. The document is kept on a conceptual level, referencing details and technical descriptions where required or advisable.

This Architecture Guide will be updated and extended in the near future. Technological trends that are foreseeable will be focused in later editions. Detailed technical descriptions and configuration examples will be part of other documents or, if applicable, an appendix to this Architecture Guide.

This first edition of the Architecture Guide emphasizes planning, deployment and operation of OpenShift Enterprise 3. This includes aspects like integrating external services, sizing and operation.

The second edition covers the identification of single points of failure during the design phase, references to implementation of a highly available OpenShift Enterprise 3 platform and additional operational aspects of high availability.

The third edition of this Architecture Guide will cover additional capabilities of OpenShift Enterprise 3.1 like central system and container logs, metrics and a web console extension to query logs and metrics.

Throughout this Architecture Guide we follow the IEC prefix notation [2: https://en.wikipedia.org/wiki/Binary_prefix#IEC_prefixes].

Introduction

This Architecture Guide will introduce the reader to ACME Corp, a company used to give examples for this Architecture Guide and follow up reference implementations. ACME Corp is a company that is reused within other Architecture Guides as well. Following a brief introduction to ACME, planning and design considerations for OpenShift Enterprise 3 are discussed. The third part of this document will elaborate on operational aspects of an OpenShift Enterprise 3 environment.

This chapter will describe:

- ACME Corp's business goals
- organizational and informational structure of the company
- roles within the organization
- requirements derived from business goals

ACME Corporation

The Acme Corporation is a fictional corporation that features outlandish products that fail or backfire catastrophically at the worst possible times.

ACME's core business is manufacturing and selling products. ACME also sells tickets for events. To support these goals, ACME hosts an online presence (their website) and an online ticket shop (as part of their website). While the website is in production, the online shop is still in development phase.

ACME Corp has made the decision to migrate all of its applications to a new platform and update their internal processes to adopt ideas from the devops and agile methodologies.

This ACME Corp initiative serves three purposes: (a) reuse components that are available within the company, (b) deploy new applications faster and (c) prepare for a higher demand of explosive IT solutions.

ACME Corp's organizational structure

Beside the obvious and mandatory organizational units (eg. CxO offices, legal, marketing, etc.), ACME's organizational structure is straight forward: an **Information Technology Department** is responsible to develop and operate IT applications so that business requirements are fulfilled. Required Infrastructure is also operated by **IT Dept.**

The **IT Operations** department is taking care of all data center management tasks and owns the entire production environment. They build and oversee all infrastructure layers, including computing (both physical and virtual), storage and network resources. These responsibilities include monitoring, incident management and backup and restore as well. In addition, they are responsible for all operating system-related items, including operating system upgrades and security patching.

The **Software Development** Department is responsible for creating custom software and integrating it with 3rd-party software components. These responsibilities include all application-specific building and testing activities and configuration management.

IT Services Management is responsible for all architecture and process areas in IT. These include Service Validation and Testing (QA), Release Management, and Security Management as the key processes.

To be better prepared for the future, ACME recently founded an **Emerging Technologies** Department to investigate upcoming trends and technologies early.

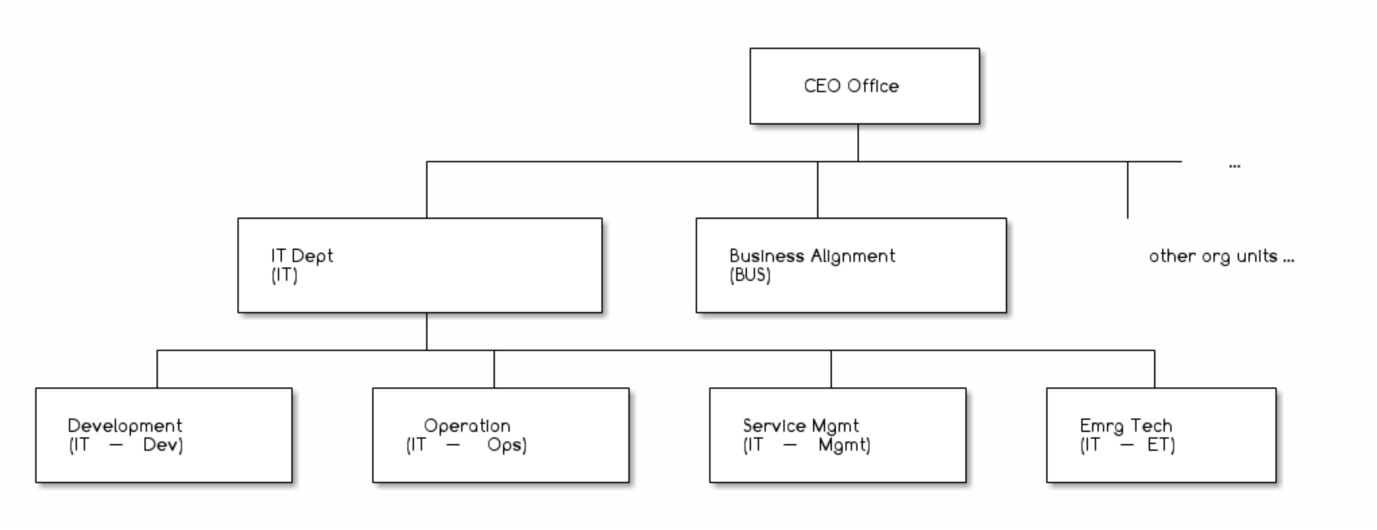

Figure 1. ACME Corp Org Chart

It is required by the Head of IT Deptartment that each organizational unit follows a strict process to gather requirements, develop software, deploy and manage the software. One of the highest priority goals of the IT Department is to **automate as much as possible** while remaining flexible enough to deliver changing needs by other departments.

'The Process' (as the process definition of the Head of the IT Department is called) implements a **three-stage environment**, consisting of 'DEV' (development), 'QA' (quality assurance) and 'PROD' (production). Nevertheless, a new initiative has been implemented and 'devops' and agile methods are gaining a broader focus within the IT Dept.

Introducing devops in a very strictly devided environment may also introduce new challenges, like new operational processes and new governance aspects.

ACME Corp’s roles

Within ACME Corp a few roles have been standardized. This is a list of roles focused on Information Technology and Business Alignment departments, other departments are out of scope for this Architecture Guide.

Table 1. ACME organizational roles

org. unit	role name	role description	responsibility
BUS	App Own	Application Owner	owns the application that supports business
BUS	Req Eng	Requirements Engineer	gather and formalize requirements of other parts of ACME to be handed over to IT-Dev
IT	Prod Own	Product Owner	owns the IT Product that implements Business Requirements
IT-Dev	Dev	A software developer	responsible for delivering software.
IT-Dev	Ops Eng	A development engineer	able to add images to the platform
IT-Dev	QA Eng	Engineer performing QA tasks	deliver QA test results and reports
IT-Ops	App Ops	Application Operator	overall responsibility to operate an application
IT-Ops	Plt Ops	Platform Operator	operates the Application Platform infrastructure
including storage provisioning	but not the SOE	IT-Ops	Ops Mgr
Operations Manager	a read-only role primarily for reporting	IT-Mgmt	OS Adm
Operating System Administrator	maintain service of an operating system instance	physical or virtual	IT-Mgmt
Sys Eng	Core Build Systems Engineer	defines the core build (operating system)	IT-Mgmt
IT Ops	Infrastructure Operator	maintain server for Network	Storage
Compute resource	IT-Sec	Sec Eng	Security Engineer

ACME Corp Datacenter Topology Overview

ACME Corp is based in Munich, Germany. The primary data center is in Munich as well. This data center has a segregated network that is used for frontend servers, which are available to the outside world. Most of the infrastructure services are running in Munich, where the entire IT operations organization is based as well.

Only the web application development and testing are based in the United States in Boston. Both locations run small data centers. Munich uses Red Hat Enterprise Virtualization as the underlying virtualization platform. Boston uses the Red Hat Enterprise Linux OpenStack Platform.

ACME Corp has considered using public cloud resource providers to implement a hybrid cloud setup but has not implemented it yet.

The following diagram illustrates the data center topology.

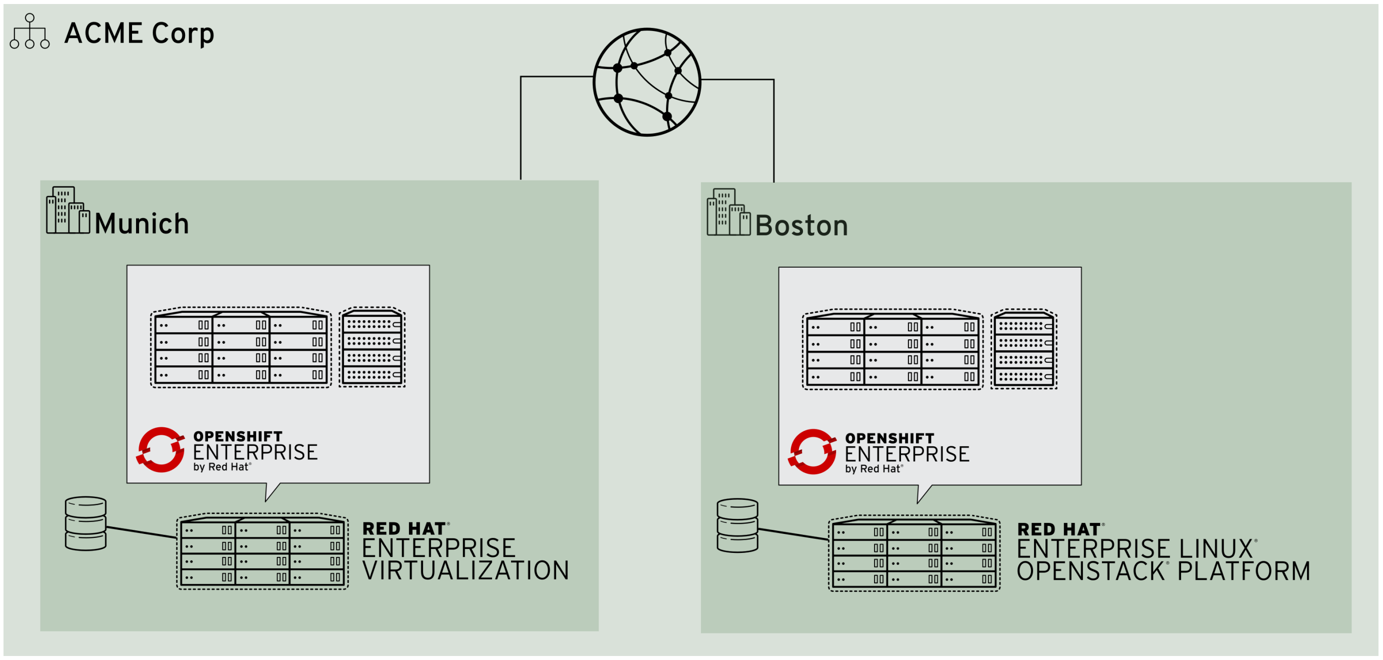
ACME Corp
Munich
OPENSHIFT ENTERPRISE by Red Hat
RED HAT ENTERPRISE VIRTUALIZATION
Boston
OPENSHIFT ENTERPRISE by Red Hat
RED HAT ENTERPRISE LINUX OPENSTACK PLATFORM

Figure 2. ACME Datacenter Topology

ACME Application Architecture Overview

Since ACME Corp has just been founded and the business model is still evolving, we currently have only one business application: ACME Website, which is the public web presence of ACME.

A second business application is currently in the design phase: the ACME Webshop.

The ACME website is based on the famous WordPress application, but the ACME webshop will become a JEE app running on top of JBoss Enterprise Application Server. Both use MySQL as database backend. To achieve a higher degree of standardization, ACME has decided to use a standardized MySQL configuration as shared component.

ACME Corp's IT requirements

The Head of IT Dept. of ACME Corp. has set a few specific requirements and goals for the department. These targets should be met by any IT project within ACME.

- deployments shall be fast - regardless of the deployment trigger, may it be a security patch or an update to online shop features or base operating system components

- developers shall reuse common components - common components shall be deployed onto a Platform as a Service and shall be reused as long as it is not proven they cannot be reused
- developers shall deliver components - these components must not be split up into readmes, source or any number of items, but be one piece of software
- development and quality assurance environments must be separated from production environment
- there shall be only one production environment
- the production environment must be separated in at least two zones to ensure service quality
- there shall always be a human task to deploy a component to the production environment
- logging shall be ensured on all levels of an application
- monitoring of an application shall be focused on user experience

Application Lifecycle

To fulfill these requirements an Application Lifecycle has been implemented:

- a Software Developer will provide a set of container images implementing a components
- a QA Engineer will ensure release quality of the image set. The engineer will be supported by a Continuous Integration Server.
- a Application Operator will deploy the newly released image set

While Software Developers and QA Engineers will use one OpenShift Enterprise 3 environment called 'DEVQA', Application Operators will work on a second OpenShift environment called 'PROD'.

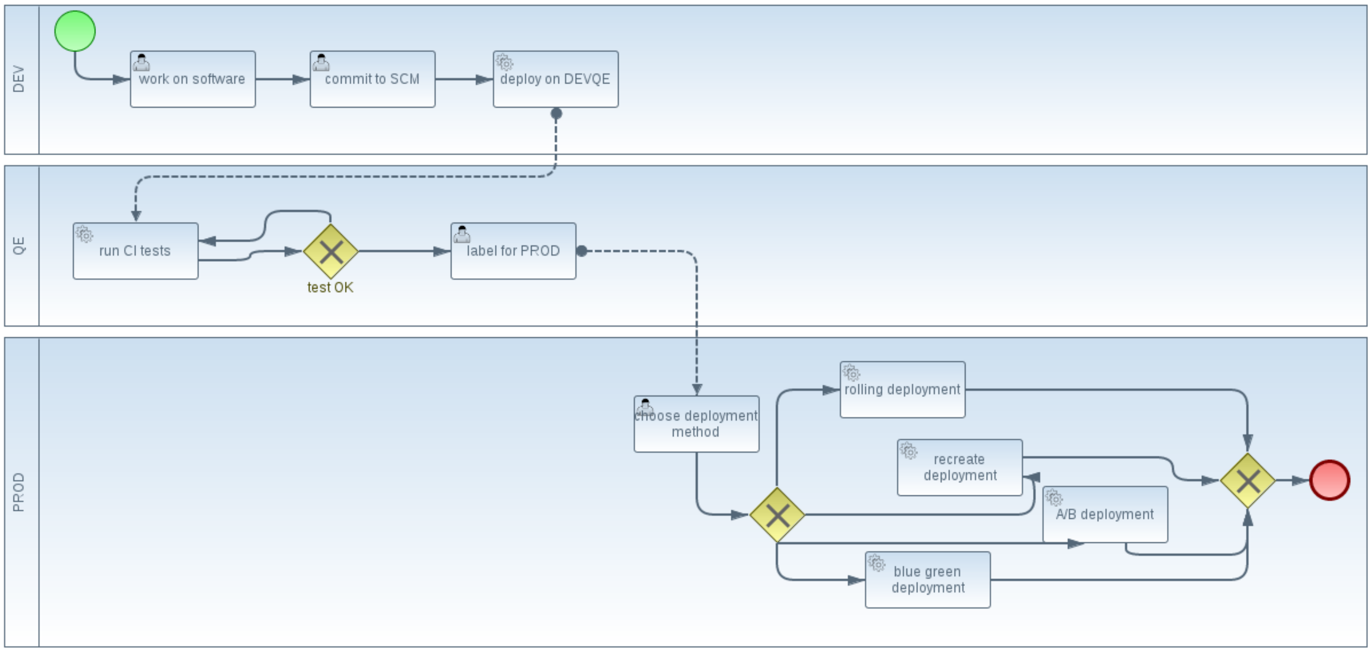

Figure 3. ACME Application Lifecycle

Standard OpenShift Enterprise 3 Components

Within OpenShift, Kubernetes manages containerized applications across a set of containers and provides mechanisms for deployment, maintenance and scaling these containers.

A OpenShift Enterprise 3 cluster consists of one or more masters and a set of nodes.

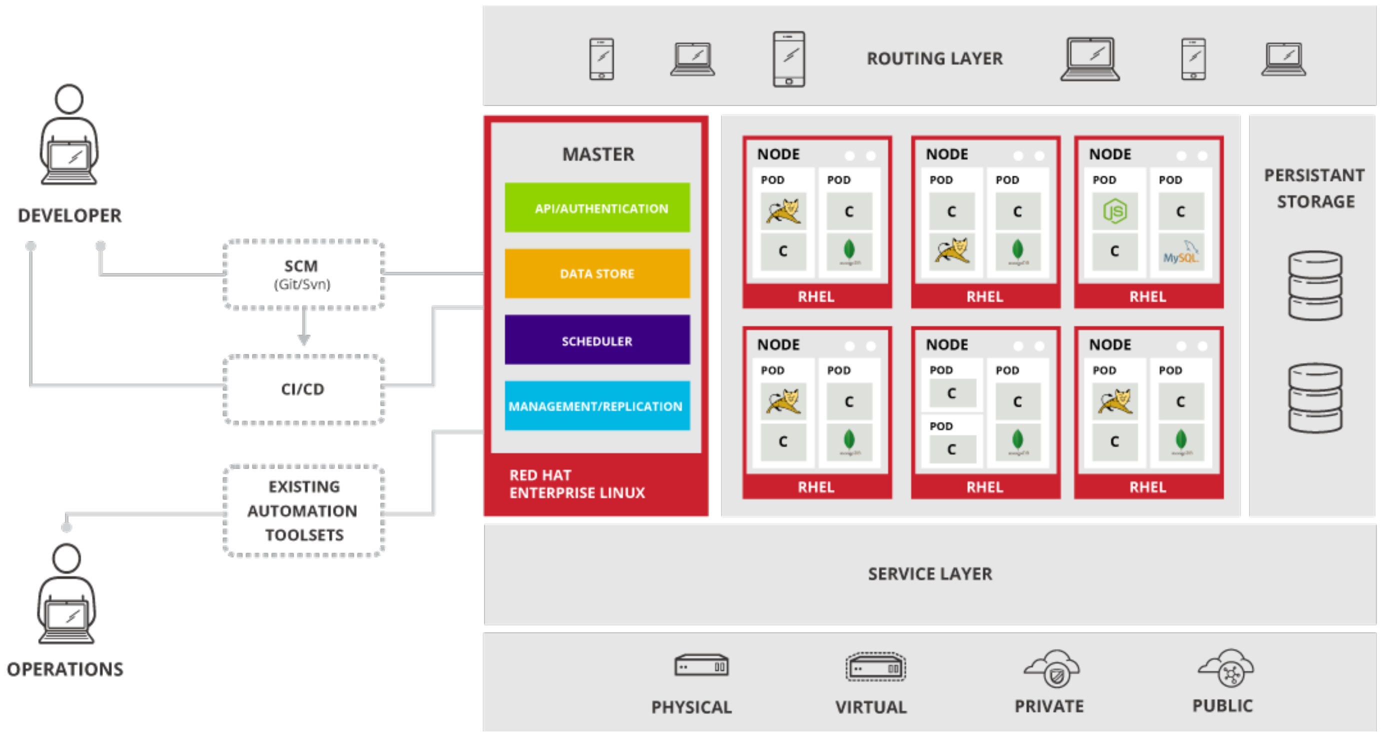

Figure 4. OpenShift Enterprise 3 Architectural Overview

Masters

The master is the set of hosts that contain the master components, including the API server, controller manager server and etcd. The master maintains the clusters configuration, manages nodes in its OpenShift cluster and schedules pods to run on nodes.

In a single master configuration the availability of running applications remains if any of the master components fail. However, failure of any of the master components reduces the ability of the system to respond to application failures or creation of new applications. You can optionally configure your masters for high availability to ensure that the cluster has no single point of failure.

A detailed description is available as Deploying an OpenShift Enterprise 3 Distributed Architecture

API Server

The Kubernetes API server validates and configures the data for pods, services and replication controllers. It also assigns pods to nodes and synchronizes pod information with the cluster configuration.

etcd

etcd stores the persistent master state while other components watch etcd for changes to bring themselves into the desired state. etcd can be optionally configured for high availability, typically deployed with 2n+1 peer services.

Controller Manager Server

The controller manager server watches etcd for changes to replication controller objects and then uses the API to enforce the desired state.

Nodes

A node provides the runtime environments for containers. Each node in a OpenShift cluster has the required services to be managed by the master.

The master uses the information from nodes to validate nodes with health checks. A node is ignored until it passes the health checks, and the master continues checking nodes until they are valid.

Kubelet

Each node has a kubelet that updates the node as specified by a container manifest, which is a YAML file that describes a pod. The kubelet uses a set of manifests to ensure that its containers are started and that they continue to run.

Service Proxy

Each node also runs a simple network proxy that reflects the services defined in the API on that node. This allows the node to do simple TCP and UDP stream forwarding across a set of back ends.

Registry

OpenShift can build Docker images from your source code, deploy them, and manage their lifecycle. To enable this, OpenShift provides an internal, integrated Docker registry that can be deployed in your OpenShift environment to manage images.

The registry stores Docker images and metadata. For production environment, you should use persistent storage for storage for the registry, otherwise any images anyone has built or pushed into the registry would disappear.

Router

Pods inside of an OpenShift cluster are only reachable via their IP addresses on the cluster network. An edge load balancer can be used to accept traffic from outside networks and proxy the traffic to pods inside the OpenShift cluster.

An OpenShift administrator can deploy routers in an OpenShift cluster. These enable routes created by developers to be used by external clients.

OpenShift routers provide external hostname mapping and load balancing to services over protocols that pass distinguishing information directly to the router; the hostname must be present in the protocol in order for the router to determine where to send it. Routers support the following protocols:

- HTTP
- HTTPS (with SNI)
- WebSockets
- TLS with SNI

OpenShift software-defined network (SDN)

OpenShift uses a software-defined networking approach to provide a unified cluster network that enables communication between containers across the OpenShift cluster. This cluster network is established and maintained by the OpenShift SDN, which configures an overlay network using Open vSwitch (OVS).

ACME Corp specific Components

This chapter will introduce some ACME Corp owned and hosted applications that need integration with OpenShift Enterprise 3.

Domain Name System

As a base for the general operation of any corporate IT, ACME Crop runs a Domain Name System (DNS) infrastructure. The DNS servers are the authoritative source for all internal DNS zones and it is using external DNS servers to forward name resolution to.

Storage System

Within ACME Corp a central storage system is providing services via Fiber Channel, iSCSI, CIFS and NFS. This storage system distributed across the data centers and its NFS service will be used by all OpenShift environments.

Corporate GIT Server

Git is a fast, scalable, distributed revision control system. The corresponding software packages are part of Red Hat Enterprise Linux, and newer versions are also available in Red Hat Software Collections.

We are using Red Hat Satellite 6.1 to provision and maintain various git servers automatically, to enable better support of our ACME software development efforts and projects. A detailed description how these Git server are set up is provided as part of [sat6-soe].

ACME Corp's Git server will be used in all OpenShift Enterprise 3 build configurations.

Continuous Integration Server

ACME Corp is using a Continuous Integration (CI) server called Jenkins to perform a continuous cycle of automated tests for their website and webshop application. The CI server is hosted as a service on a separate host and should be integrated into the DEVQA environment.

Automated testing should be performed. According to the Application Lifecycle a human task shall be carried out to promote a new version of the webshop from DEVQA to PROD.

Monitoring and Alerting System

For the purpose of Application, Platform and Infrastructure monitoring and alerting ACME Corp is maintaining a Zabbix Monitoring system. This has been integrated with the Standard Operating Environment. Further details are provided in [sat6-soe].

Systems Management Infrastructure

The OpenShift Platform will be installed on top of Red Hat Enterprise Linux. To maintain an up-to-date and secure infrastructure, it is absolutely necessary to actively manage all components of OpenShift. A highly automated systems management infrastructure is recommended.

OpenShift itself is infrastructure agnostic: it may be running on bare metal, on a virtual infrastructure such as Red Hat Enterprise Virtualization 3.5 or on top of an OpenStack environment.

As ACME Corp has one Red Hat Enterprise Virtualization 3.5 and one Red Hat Enterprise Linux OpenStack Platform 7 based environment, we will have a short review of the requirements for both environments. In addition to these infrastructure components, a Red Hat Satellite 6.1 will be used to provide a Standard Operating Environment.

Role of Red Hat Enterprise Virtualization

Red Hat Enterprise Virtualization 3.5 will provide the (virtualized) hosts to run the OpenShift platform. Networking, Storage and Compute capabilities will be managed by Infrastructure Operators and build the foundation for OpenShift. On top of this infrastructure Operating System Administrator will provide an Operating System and maintain its configuration and keep it up to date.

It is advisable to preconfigure compute profiles so that these can be reused to deploy new hosts for the OpenShift infrastructure easily.

Role of Red Hat Enterprise Linux OpenStack Platform

Red Hat Enterprise Linux OpenStack Platform 7 provides Infrastructure-as-a-Service (IaaS) foundation for public, private or hybrid cloud computing environment on top of Red Hat Enterprise Linux. Red Hat Enterprise Linux OpenStack Platform 7 meets enterprise requirements with ability to extensively scale and provide a fault tolerant and highly available environment. OpenStack is made up of many different moving parts. Because of its open nature, anyone can add additional components to OpenStack to meet their requirements. The Red Hat Enterprise Linux OpenStack Platform IaaS cloud is implemented by a collection of interacting services that control its computing, storage, and networking resources. Infrastructure Operators and Operating System Administrator will provide an infrastructure and Operating System base and maintain its configuration and keep it up to date.

Before using Red Hat Enterprise Linux OpenStack Platform 7 a review of 'images' and 'security groups' should be conducted to meet OpenShift Enterprise 3 requirements.

Role of Red Hat Satellite 6.1

Red Hat Satellite 6.1 is a system management solution that makes infrastructure easier to deploy, scale, and manage across physical, virtual, and cloud environments. It enables Infrastructure Operators and Operating System Administrator to provision, configure, and update hosts to help ensure that they are running efficiently and securely.

A Standard Operating Environment

Standardized operating environments help business operate effectively and efficiently. Red Hat solutions like Red Hat Enterprise Linux OpenStack Platform 7 and Red Hat Enterprise Virtualization 3.5 provide additional benefits, such as improved total cost of ownership (TCO), better IT productivity, and increased agility. Red Hat Enterprise Linux and Red Hat Satellite 6.1 provides an ideal platform for standardized operating environments with lower TCO and greater IT efficiency and productivity, adding to the bottom line and increasing business agility and competitiveness.

Design your OpenShift Enterprise 3 infrastructure

This chapter will introduce the Use Cases we will describe in detail using the ACME Corp example. An overview of use cases including a short description will be given, followed by OpenShift design recommendations concluded from the use cases.

This Architecture Guide focus on the infrastructure agnostic nature of OpneShift: it does not imply any particular infrastructure. OpenShift Enterprise 3 can be deployed on top of physical or virtual infrastructure, on top of a virtualization environment, on a private cloud or a public cloud infrastructure.

Use Cases covered by this Architecture Guide

This chapter will list the basic use cases addressed within this Architecture Guide to fulfill ACME Corp requirements.

OpenShift Enterprise 3 on Red Hat Enterprise Virtualization 3.5 with Red Hat Satellite 6.1

We will focus on deploying OpenShift Enterprise 3 based on a virtualized Infrastructure provided by Red Hat Enterprise Virtualization 3.5. In this environment Red Hat Satellite 6.1 will be used to deploy a standard operating environment, OpenShift's advanced installation method will be used to deploy and maintain OpenShift components and configurations.

Alternatively a deployment using OpenStack will be described by [osp-aep].

Using Red Hat Satellite 6.1 to deploy onto Red Hat Enterprise Virtualization 3.5 enables ACME Corp to use it's infrastructure in a very efficient way, giving opportunities to scale the OpenShift Platform and enables ACME Corp to segregate most of the duties to different roles.

OpenShift Enterprise 3 on a virtual infrastructure will also enable easy and fast updates of the whole software stack. Roles and responsibilities segregated by infrastructure, platform and application enable the shortest possible turnaround time for each layer to be updated.

A design guide to OpenShift Enterprise 3

OpenShift Enterprise 3 requires you to plan some aspects of your infrastructure upfront. This includes hardware sizing, network design and storage provisioning. The following chapters will give a overview and good practice based recommendations to these topics.

Hardware sizing

As with any software installation, OpenShift requirements regarding CPU and Memory are highly dependent on the workloads themselves. In addition to the software requirements, OpenShift components themselves require at least 8GB of RAM. Each host (physical or virtual) should have at least 2 CPU cores. OpenShift nodes may benefit from more CPU cores, as the containerized workloads may utilize them in a more efficient way.

Storage recommendations

In addition to the disk usage for the base operating system and OpenShift packages, storage for the master components (like etcd), docker images on each node, central storage to be provisioned to pods (via PersistantStorageClaims) and storage for central logging is required.

As of OpenShift Enterprise 3.0 the only option to provide storage to pods is via NFS [3: https://access.redhat.com/documentation/en/openshift-enterprise/version-3.0/openshift-enterprise-30-whats-new/chapter-2-openshift-enterprise-30-release-notes] more options are available as technology preview and will be included in later releases.

Table 2. OpenShift Enterprise 3 storage recommendations

Component	Size	Comment
Base Operating System	12Gi	RHEL Server components
Master incl. etcd	16Gi	to store configuration of the OSE3 environment
central logging	24Gi	for consolidated system logs
Node’s docker storage	24Gi	this will be used by docker containers
central storage	24Gi	to be provided as PersistentVolumes to Pods via NFS

Even though GPT [5: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Installation_Guide/appe-disk-partitions-overview.html#sect-disk-partitioning-guid-partition-table] is used to partition your hard drives, you must not use more than 128 partitions per hard disk.

Providing Storage to Nodes

Each node in a OpenShift cluster requires a certain amount of disk capacity to be used by containers while they run, or to be used for container images that have been pulled from a registry. From the point of view of the operating system, this storage is local storage: either a disk device of a virtual host in Red Hat Enterprise Virtualization 3.5 or a volume attached to a Red Hat Enterprise Linux OpenStack Platform 7 instance.

Providing Storage to Pods

Storage can be requested by a Pod via `PersistentVolumeClaim`. To satisfy this request/claim OpenShift will match the claim against a pool of available volumes. To provide storage in the most efficient way the pool should container many volumes of different sizes. A good ratio of percentage of volumes and sizes is displayed in the following table.

Given, for example, (a) a total of 128Gi and as (b) a second example 1Ti of storage available for provisioning will result in the number of partitions shown below.

Table 3. OpenShift Enterprise 3 number of volumes and size

Volume Size	Percentage of Volumes	Example (a): number of partitions	Example (b): number of partitions
1Gi	20	23	186
5Gi	50	11	93
10Gi	25	3	23
20Gi	15	1	7

Take a look at example (b), the number of 1Gi partitions is 186, which is not allowed, as the maximum number of partitions is 128.

Either way, you need to review and forecast the storage requirements of the application workloads you will deploy to OpenShift Enterprise 3.

To configure a RHEL7.1 based NFS server an Ansible playbook is provided with the advanced installation method.

Regions and Zones

Regions and Zones represent the concept of affinity and antiaffinity in OpenShift. These concepts could, for example, be used to implement resource placement algorithms. 'Regions' are most often used to allocate resources on two or more data centers, so that the availability of a service could be increased.

In fact, OpenShift is topology agnostic, so regions and zones are just one concept that we use to think about infrastructure topologies. OpenShift Enterprise 3 provides advanced controls for implementing any topology desired, leveraging filtering and affinity rules to ensure that resources of an applications are either grouped together (region) or spread apart (zone).

The scheduling component of the openshift-master uses a list of predicated and a list of priorities to schedule the instantiation of a pod somewhere on the cluster. Given all the nodes in a OpenShift cluster, 'predicates' filter out a list of candidates where a pod could be instantiated. After the filtering 'priorities' are applied, select the candidate list a little further.

In conjunction with Node labels and the node’s own NodeSelector, complex algorithms to determine pod placement could be implemented.

For the ACME Corp we will implement two separate OpenShift Enterprise 3 environments: `devqa` and `prod`.

For each environment we will implement at least two regions:

- `services` - for routers and the registry
- `routers` - for routers in production environment
- `devqa` or `prod` - for applications

For the `prod` region we will implement two zones `us` and `eu`.

The following two diagram will give you a graphical overview of DEVQA and PROD.

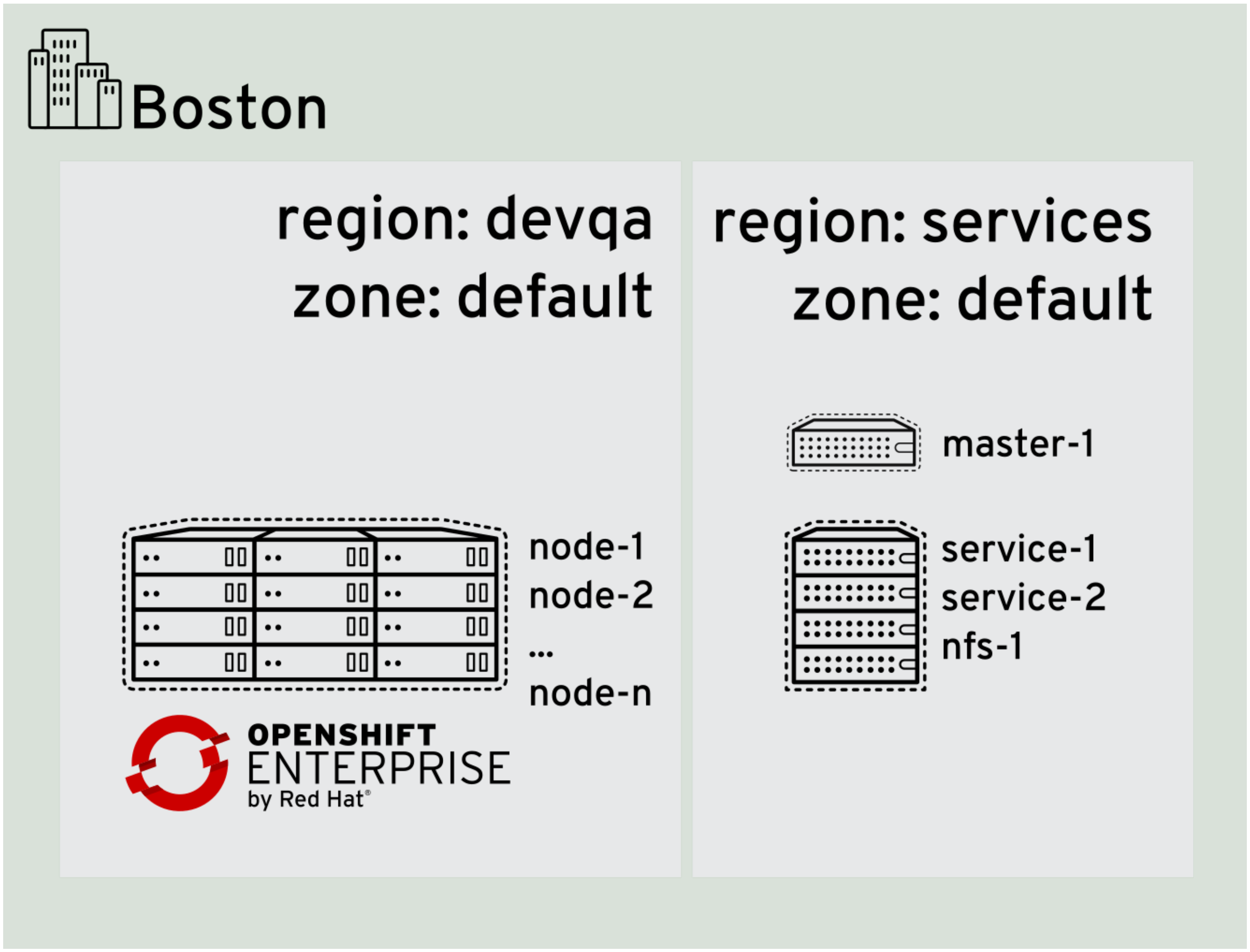

Figure 5. ACME DEVQA Components, Regions, Zones

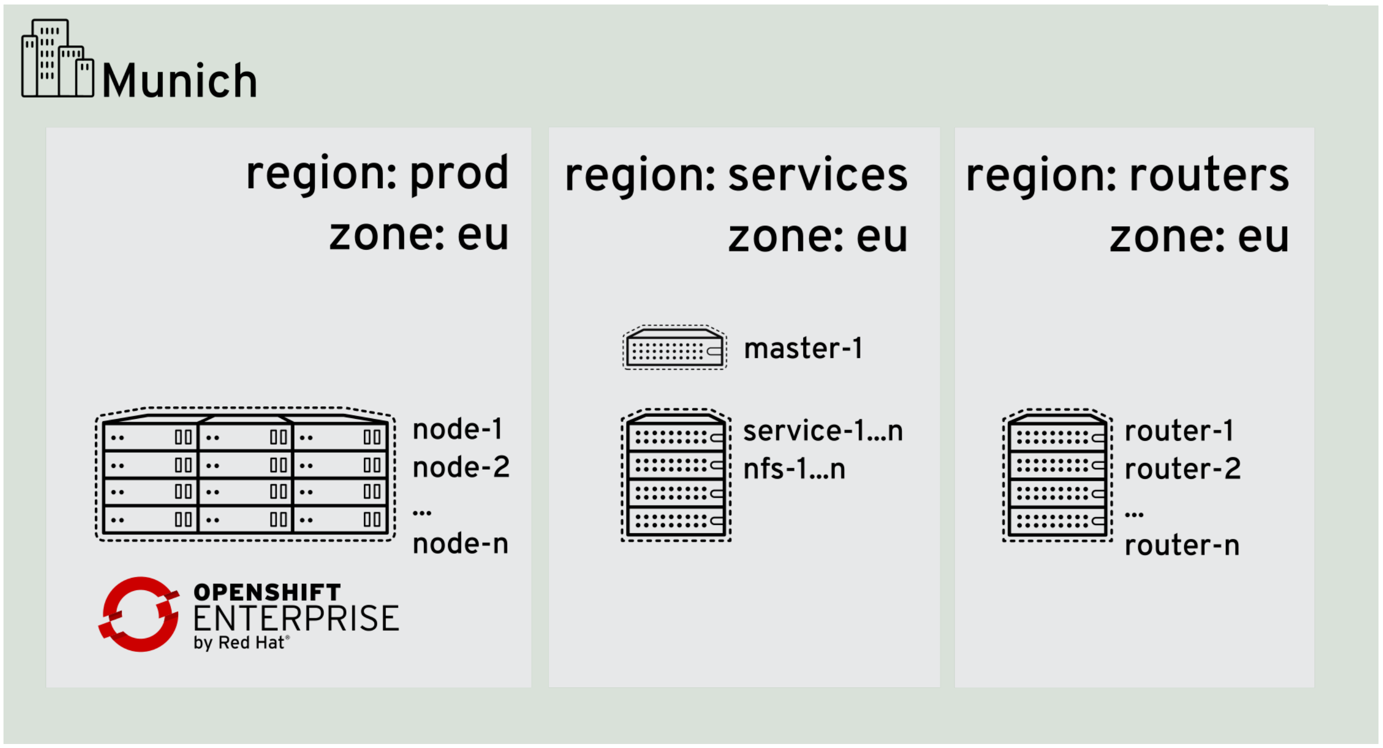

Figure 6. ACME Corp PROD components, Regions, Zones

Registries

The OpenShift integrated registry and all routers will be run within the 'services' region. We recommend configuring any registry with persistent storage, so that images uploaded to the registry will not disappear in the event of redeploying a registry. Further more we recommend backing up the registries.

Operational responsibility for the OpenShift integrated registry is owned by the Platform Operator.

Red Hat Satellite 6.1 as a third party registry

OpenShift Enterprise 3 could be configured to use any third party registry. We recommend that you use Red Hat Satellite 6.1 as a gatekeeper: it will download images from third party registries and act as a single source of trust and a control point to filter registry's content.

We recommend that the OpenShift integrated registry is always secured by using TLS [6: https://access.redhat.com/documentation/en/openshift-enterprise/version-3.0/openshift-enterprise-30-administrator-guide/chapter-1-installing#securing-the-registry].

Routers

The OpenShift router is the ingress point for all external traffic destined for services in an OpenShift environment.

Highly available Routers

Highly available router configuration can be accomplished by running multiple instances of the router pod and deploy a load balancing tier in front of them. This could be a DNS round robin. This pattern is described in production environment in the DNS considerations chapter.

Securing Routers

Creating a secure route [7: https://access.redhat.com/documentation/en/openshift-enterprise/version-3.0/openshift-enterprise-30-architecture/chapter-3-core-concepts#secured-routes] to the pods can be accomplished by specifying the TLS Termination of the route and, optionally, providing certificates to use.

It is recommended to use 'Edge termination': TLS termination occurs prior to traffic reaching the destination. TLS certificates are served by the frontend of the router.

DNS considerations

Within the OpenShift platform, the concept of 'routes' is used as a way to expose services to the outside world. By mapping a hostname to a service, a route will make an OpenShift hosted application accessible.

A defined route and the endpoints identified by its service can be consumed by a router to provide named connectivity that allows external clients to reach your applications.

For this concept to work you will need to have a wildcard for a DNS zone to resolve the IP address(es) of the OpenShift router.

To keep the set of IP addresses where a router may be deploy predictable and relatively small, we recommend running the routers in the 'infra' region of your OpenShift environment. Setting up the wildcard DNS zone resource record to point to all IP addresses of the hosts of the 'infra' region will satisfy the DNS requirement.

Looking at ACME Corps DEVQA environment this could lead to the following DNS configuration:

Table 4. OpenShift Enterprise 3 DNS design

DNS domain name	pointing to	environment	region	zone
`*.devqa-apps.acme.example.com`	`infra-1.devqa.acme.example.com`, `infra-2.devqa.acme.example.com`	DEVQA	infra	default
`*.apps.acme.example.com`	`routers.eu.prod.acme.example.com`	PROD	infra	eu
`*.apps.acme.example.com`	`routers.us.prod.acme.example.com`	PROD	infra	us
`routers.eu.prod.acme.example.com`	`infra-1.eu.prod.acme.example.com`, `infra-2.eu.prod.acme.example.com`	PROD	infra	eu

DNS domain name	pointing to	environment	region	zone
routers.us.prod.acme.example.com	infra-1.us.prod.acme.example.com, infra-2.us.prod.acme.example.com	PROD	infra	us

If you are going to implement a geography-aware DNS service, it should make a decision based on the requesters location to either response with either `routers.eu.prod.acme.example.com` or `routers.us.prod.acme.example.com` if queried for `*apps.acme.example.com`.

A typical request flow for a end user browsing the ACME Corp Ticket Shop could look like this:

- a German user is requesting the ticket shop at 'tickets.apps.acme.example.com'
- the user's DNS resolver will answer with 'routers.eu.prod.acme.example.com'
- a web request is made to that host
- the request will be forwarded by one of the router's pods to the service

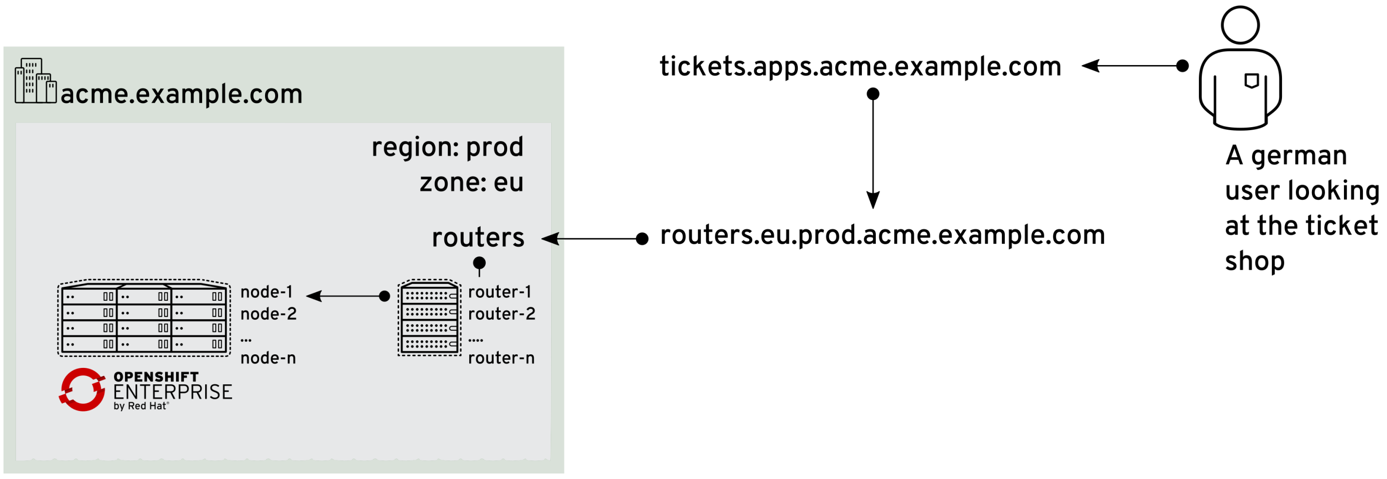
acme.example.com
region: prod
zone: eu
routers
node-1
node-2
...
node-n
router-1
router-2
....
router-n
OPENSHIFT ENTERPRISE by Red Hat
tickets.apps.acme.example.com
routers.eu.prod.acme.example.com
A german user looking at the ticket shop

Figure 7. ACME Ticket Shop request flow

Of course, ACME Corp would typically alias the hostname structure behind another DNS entry like 'shop.acme.example.com' pointing to 'tickets.apps.acme.example.com'.

Data Security

Data security aspects should be considered while designing any IT solution. For OpenShift Enterprise 3 there must be a clear focus on

- data protection
- backups

Data Protection

To protect data used or hosted by the OpenShift platform we need to identify where data is stored and how it is transfered.

Given the OpenShift architecture there are only two places where data is stored

a. the central NFS server providing storage to pods and
b. the openshift-master component etcd holding configuration data of OpenShift itself.

In most cases data is transfered encrypted via https, this is true for all OpenShift components communicating between each other. Exposing an OpenShift service via http is possible but not encouraged.

Data is transfered unencrypted between the NFS server and all nodes running a Pod. As a pod may require persistent storage, the host that is running the pod will mount some NFS export from the NFS server, NFS in an OpenShift platform will use unencrypted transfer.

Data Security and Backups

Now that all point within the OpenShift platform that store data are identified, it is important to set up backup and restore procedures for this storage points.

Which and how this should be handled is described in further detail in [Backup and Restore].

Network segregation considerations

In a default environment (even when using the advanced installation method) OpenShift Enterprise 3 is using one network for all its communication. Obviously this network could and should be segregated: network traffic that is 'internal' (e.g. kubelets updating their status, or etcd queries) should be on an 'internal' network.

ACME Corp DEVQA design

Within this chapter we will apply the design guide to ACEM Corp’s DEVQA environment. The goal is to design a environment which can be used by ACME’s QA Eng staff to test, assure and approve the quality of a software release. According to the application lifecycle the new and tested release shall be tagged for production.

Overview of ACME Corp DEVQA infrastructure

For the DEVQA environment one separate installation of OpenShift Enterprise 3 will be deployed. It will provide a docker registry (as an OpenShift integrated registry) with persistent storage. Three regions will be configured: 'devel', 'devqa' and 'service', each region has one zone: 'default'.

DEVQA will be deployed on top of Red Hat Enterprise Virtualization 3.5. A central storage component is used: each virtual host will have iSCSI attached disks and a NFS server based on RHEL 7Server will provision storage to pods in all zones. No network segregation will be configured.

Routers and the NFS Server and the registry will be deployed to the 'service' region. Any pod deployed for any project of [IT-Dev-Dev] will be deployed to the 'devel' or 'devqa' zone.

The [Architectural Overview of ACME Corp DEVQA environment] gives an overview of DEVQA.

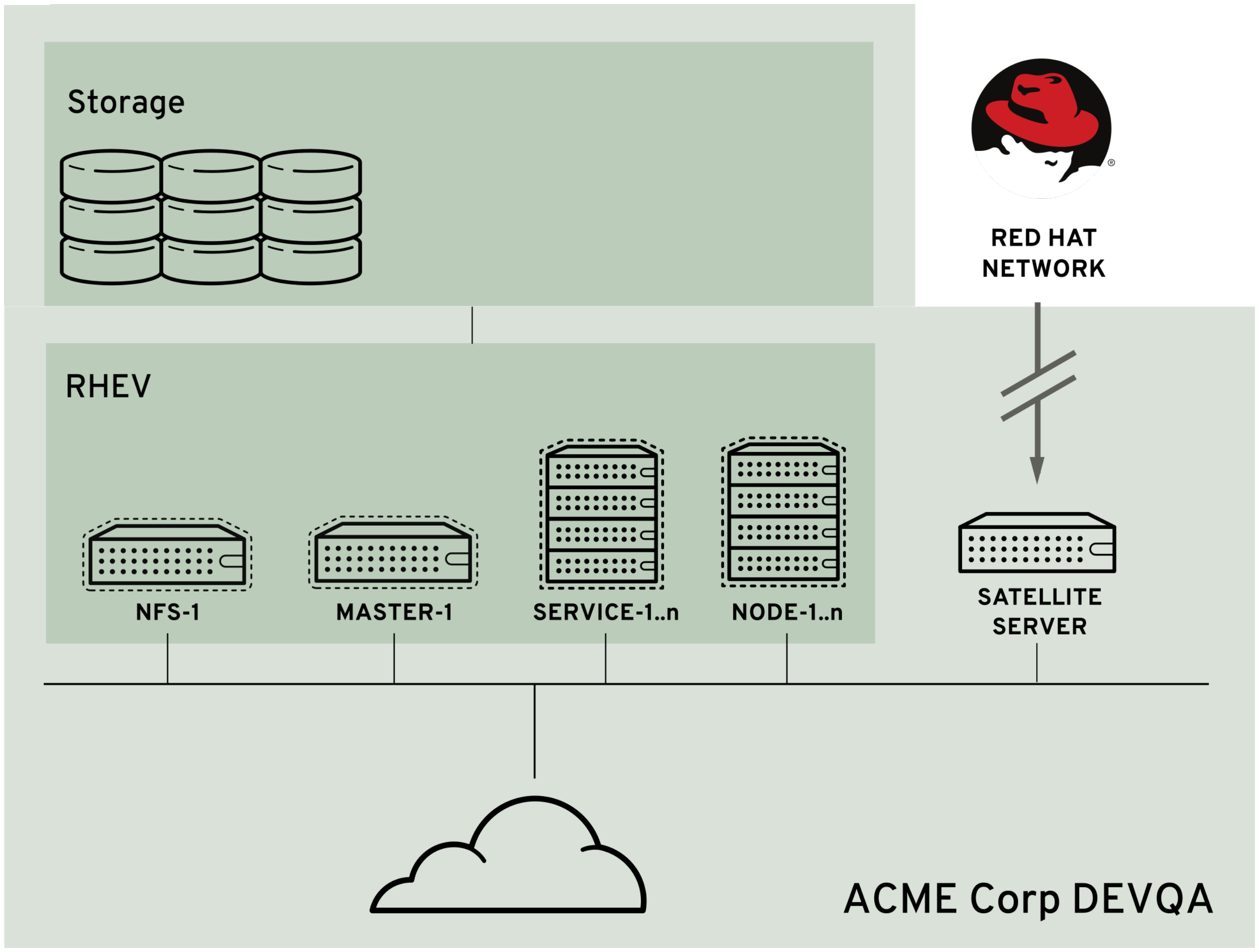

Figure 8. Architectural Overview of ACME Corp DEVQA environment

According to the DEVQA component model ACME Corp will use a Red Hat Satellite 6.1 to deploy and manage a set of hosts (implemented as VMs on Red Hat Enterprise Virtualization 3.5). Storage is provided by a third party storage system and is not subject to further detailing. Each host will be deployed and managed using the SOE pattern described in detail by [sat6-soe]. These hosts will be the base for a advanced installation of OpenShift Enterprise 3.

ACME Corp PROD design

In addition to ACME Corp DEVQA design this chapter adds recommendations related to the OpenShift PROD environment.

In general the same design pattern as for DEVQA is applied. For the PROD environment one separate installation of OpenShift Enterprise 3 will be deployed.

It will provide an highly available OpenShift integrated registry with persistent storage. This registry will only contain images that have been tagged as 'prod' by QA Engineering, so that the application lifecycle will be implemented.

Self-Provisioning of projects should be disabled in the PROD environment.

Two regions will be configured: 'prod' and 'service', each region has two zones: 'us' and 'eu'. OpenShift nodes labeled to be in zone 'eu' will be deployed on top of Red Hat Enterprise Virtualization 3.5 while nodes labeled to be in zone 'us' will be deployed on top of Red Hat Enterprise Linux OpenStack Platform 7.

A highly available and distributed NFS server based on RHEL 7Server will provision storage to pods in all zones. No network segregation will be configured.

Routers and NFS Servers and the registry will be deployed to the 'service' region. Any pod deployed for any project of [BUS-App-Own] will be deployed to the 'prod' zone.

Integration Points

OpenShift Enterprise is not a stand-alone solution. It is most beneficial if fully integrated with the rest of (existing) IT infrastructure. While designing the OpenShift Enterprise platform, ACME Corp has decided that the user should get a 'single sign on' (SSO) experience. Therefore the OpenShift Enterprise environment shall be integrated with a number of other services.

The following chapter will describe these integration pattern and point to more detailed resources that could be used to implement these integrations.

Integration with Red Hat Satellite 6.1

OpenShift Enterprise 3 should be integrated with Red Hat Satellite 6.1 so that

- an Infrastructure Operator could easily scale the infrastructure used by the platform
- an Infrastructure Operator could provide the inventory file for the advanced installation method
- a Platform Operator could perform configuration of the platform
- Red Hat Satellite 6.1 could be used as ingestion point for Red Hat provided images, like RHEL7, EAP 6.4 etc.

Red Hat Satellite 6.1 will be used to maintain the infrastructure, deploy updates, perform base configuration.

Red Hat Satellite 6.1 will act as a 'proxy' for container images, it will pull down all required images from Red Hat or third party registries and host them locally. Defining and providing these images shall be done by a development engineer.

OpenShift Enterprise 3 will be maintained from a single point (for ACME Corp that will be a container running the advanced installation method) that is highly integrated with Red Hat Satellite 6.1.

Configuration Management of OpenShift Enterprise 3 components

Configuration Management of OpenShift components is split in two layers:

- the core build: Red Hat Enterprise Linux on each host
- OpenShift itself: ontop of the core build

While infrastructure management and configuration management of the core build remains a task completely operated by Red Hat Satellite 6.1, the OpenShift layer utilized the advanced installation method to manage and configure components.

All configuration management task should be performed by the Platform Operator.

Identify Management (IdM)

The OpenShift master includes a built-in OAuth server. This OAuth server is used by all other components to authenticate themselves. Developers and administrators obtain OAuth access tokens to authenticate themselves to the OpenShift master (via it’s API). The Platform Operator should configure the OAuth server to use a backend service for authentication. In OpenShift this is called an identity provider.

ACME Corp will integrate the central Identity Management service hosted by a cluster of Red Hat Enterprise Linux Server. Please refer to Linux Domain Identity, Authentication, and Policy Guide As ACME Corp not only uses LDAP but also Kerberos. The DEVQA environment will be configured to use the 'RequestHeaderIdentityProvider'.

An IdM server could also be used to synchronize the OpenShift groups to LDAP groups, and further map these LDAP groups to OpenShift roles.

Integration with Identity Management services

Within the DEVQA environment OpenShift will use the 'Authentication Proxy Pattern' to authenticate users. Using this pattern will enable the addition of IdM services while OpenShift must not be reconfigured. The same pattern can be reused in the PROD environment using a different set of authentication proxies.

Configuration of OpenShift’s identify provider [8: https://access.redhat.com/documentation/en/openshift-enterprise/version-3.0/openshift-enterprise-30-administrator-guide/chapter-7-configuring-authentication#RequestHeaderIdentityProvider] shall by performed by the Platform Operator role, while management of the authentication proxy shall be performed by an Infrastructure Operator.

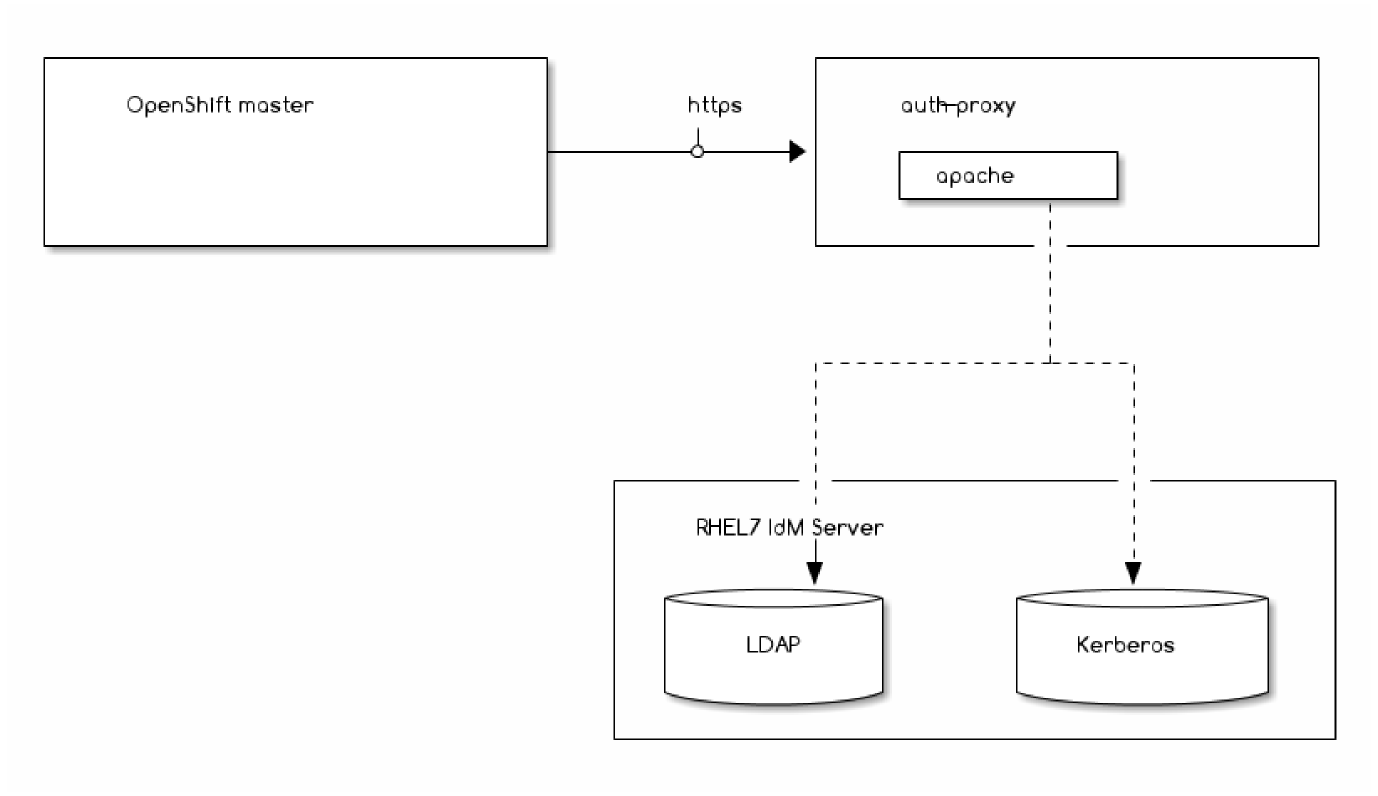

Figure 9. OpenShift integration with IdM

Integration with external (ACME Corp owned) services

Integrating external services with OpenShift Enterprise 3 hosted applications is a straight forward task. Any service that is external to the OpenShift platform, could be defined as a service within OpenShift, as long as it is reachable via predefine list of IP:PORT combinations.

In addition to services hosted (and owned) by ACME Corp itself, services that are provided 'in the cloud' via other SaaS providers could be integrated too.

The following chapters will give a few examples how to integrate external onsite services and external SaaS services.

Database Services

ACME Corp is hosting a set of Microsoft MS SQL Databases. These are provided by MS SQL AlwaysOn Failover Cluster. To integrate any database, we need to configure:

- a service within an project that may access to databases
- and endpoint for that service
- credentials used to access the endpoints

For an external service, the `selector` needs to be left blank.

The application is responsible for reading the endpoint data and credentials for the service from it’s environment and establishing a connection with the service.

Integration with Cloud services (other SaaS)

Integrating external SaaS with an application hosted on OpenShift Enterprise 3 is architectural equivalent to the approach shown in Integration with external (ACME Corp owned) services. Endpoint may be a list of FQDN and an API token may be part of the provided credentials. But the architecture and implementation does not change.

A more uncommon scenario is elaborated in the next chapter: hosting data within ACME Corp and using frontend services hosted in the cloud to access them.

Imagine a scenario where,

- either raw, unfiltered data shall not leave the company’s network, but data visualization shall happen within a cloud hosted infrastructure,
- or a set of sensors delivers huge amounts of data to an application hosted within ACME Corp.'s network.

In general we assume that all Cloud service provider offer the same capabilities like establishing a direct network connections to ACME Corp’s data centers or using any TCP/IP based protocol to communicate with ACME Corp’s data centers.

Integrating a public cloud with OpenShift Enterprise

Both scenarios described in Integration with Cloud services (other SaaS) could be implemented using 'port forwarding' [9: https://access.redhat.com/documentation/en/openshift-enterprise/version-3.0/openshift-enterprise-30-architecture/chapter-4-additional-concepts#port-forwarding].

The first scenario could, but should not, be implemented using a port forwarding to a database service hosted on OpenShift Enterprise 3. Offering a database service via port forwarding will open access to that service to all clients. Access must be regulated via firewall rules on the access network or within the OpenShift Enterprise 3 infrastructure.

Setting up a router that will allow queries to MongoDB (hosted on Openshift) from an Azure-hosted front-end service requires port forwarding from that router to the pod providing MongoDB.

setting up a database proxy as a service and exposing this to the outside cloud is more secure and should be favored over a simple port forwarding.

Shared Messaging Systems as a service of the platform

Beside the integration of existing databases, the integration of message queue systems is a common requirement.

Two scenarios could be considered:

1. integrate an external message queuing system as an external service
2. provide a shared service within the platform and 'link' it to an external message queueing system

While the first scenario's implementation follows the Database Services pattern, the second scenario will benefit more from OpenShift's management and scaling capabilities.

A project to host the shared service

For the shared messaging service it is recommended to set up a separate project to host all messaging components. In addition to that service accounts and secrets must be provided to allow access to this project from other projects.

Within this 'shared messaging service' project a JBoss A-MQ xPaas will be used to provide AMQP, STOMP and MQTT interfaces to the messaging system.

A highly available router will provide interfaces to the outside world, so that the messaging service could not only be used by applications hosted on OpenShift itself, but also by distributed client (e.g. IoT clients all over the world).

Platform considerations

If the shared messaging system shall be exposed to the outside, a dedicated set of nodes is recommended. These nodes will only run the highly available routing layer for the messaging system. Only TLS-enabled routes should be exposed on this set of routers, either as 'pass-through' or as 'edge terminated' routers.

In any case, service accounts need to be configured so that all projects allowed to access the shared messaging service are able to access it. Additionally secrets must be provided for the borking service.

Implementation

As part of JBoss xPaas images an A-MQ container image is provided. This could be used to implement the shared messaging system. The routing layer could be implemented by a customer router image, so that only the protocols are routed that should be exposed to the outside world.

Deploying OpenShift Enterprise 3

ACME Corp will use Red Hat Enterprise Virtualization 3.5 to deploy it’s OpenShift 3 Enterprise production environment in Munich and Red Hat Enterprise Linux OpenStack Platform [10: https://access.redhat.com/documentation/en/red-hat-enterprise-linux-openstack-platform/version-7/architecture-guide] in Boston for development and quality assurance.

A Red Hat Satellite 6.1 will be used for systems management: deployment of new hosts, updates of existing hosts.

Installation and configuration of OpenShift Enterprise 3 itself will be done via the advance installation method [11: https://access.redhat.com/documentation/en/openshift-enterprise/version-3.0/openshift-enterprise-30-administrator-guide/chapter-1-installing#advanced-installation] of OpenShift 3. We suggest using a container to host the OpenShift 3 administration environment. This environment will carry the latest version of the OpenShift 3 Ansible playbooks [12: https://github.com/openshift/openshift-ansible].

This infrastructure architecture illustrated by the ACME Corp example show the infrastructure-agnostic nature of OpenShift Enterprise 3: given a network and host system infrastructure, we can utilize it to provide a Platform to our developers and applications.

Prepare your Red Hat Enterprise Virtualization

Using Red Hat Enterprise Virtualization 3.5 to provide a virtualized infrastructure for OpenShift should be done via Red Hat Satellite 6.1. It will help you creating new hosts and provision required software and configuration on these hosts. Overall storage and networking resources should be derived from Hardware sizing.

Please make sure that your Red Hat Enterprise Virtualization 3.5 infrastructure is connected (via a 'Compute Resource' configuration) to your Red Hat Satellite 6.1.

Configure Red Hat Satellite 6.1 to deploy OpenShift 3 hosts

Red Hat Satellite 6.1 could be used to deploy OpenShift onto bare metal or virtualized hosts. We will describe the latter using Red Hat Enterprise Virtualization 3.5.

To use Red Hat Satellite 6.1 as your software management tool it is important to understand three of its concepts:

Entitlement for Software Repositories and Errata

Repositories (and errata is just a special case of a repository) represent the basic channel which delivers sets of software from Red Hat to the customer. To get access to OpenShift Enterprise software you have purchased a fitting entitlement. Red Hat Satellite 6.1 should have enabled the 'Red Hat OpenShift Enterprise 3.0 RPMs x86_64 7.1' repository.

Composing a Content View

A content view is a managed selection of content that contain one or more repositories and that allow optional filtering. We use them to aggregate all the repositories required for a OpenShift Enterprise 3 deployment. We recommend the good practice to use either a 'all in one' or a 'server type specific' content view for OpenShift Enterprise 3.

Please also refer to the "Content View Naming Conventions" chapter of [sat6-soe].

Defining Lifecycle Environments

Content views are 'life cycled' (or pushed forward) through defined life cycle environments. Lifecycle environments are used to separate different stages of a greater context environment. Within the ACME Corp PROD environment we still would separate the operation by versioning the content views, and pushing them forward through the lifecycle environments. For ACME Corp PROD environment we have chosen a pattern called 'Deviant Lifecycle Paths' (as described in [sat6-soe]).

Verify Red Hat Satellite 6.1 Content life cycle

We assume your have purchased an OpenShift Enterprise 3 subscription and have access to the corresponding repositories and have enabled and sync them within Red Hat Satellite 6.1.

Within Red Hat Satellite 6.1 you should have a content view named 'cv-paas-ose3', associated with the lifecycle environment 'DEV' and it should be promoted. All repositories of the following table should be included in the content view 'cv-paas-ose3'.

Table 5. cv-paas-ose3 content view

Repository Name	Repository ID	comment
Red Hat Enterprise Linux 7 Server (RPMs)	rhel-7-server-rpms	the base operating system
Red Hat Enterprise Linux 7 Server - Extras (RPMs)	rhel-7-server-extras-rpms	additional RPMs e.g. docker
Red Hat Enterprise Linux 7 Server - Optional (RPMs)	rhel-7-server-optional-rpms	additional RPMs e.g. docker
Red Hat OpenShift Enterprise	rhel-7-server-ose-3.0-rpms	OpenShift3 RPMs
Red Hat Enterprise Linux High Availability for RHEL Server	rhel-ha-for-rhel-7-server-rpms	RPMs required for HA master configurations
Red Hat Satellite Tools 6.1 for RHEL 7 Server RPMs x86_64		Red Hat Satellite 6 Tools for RHEL7

Do not include your Kickstart repositories in a content view.

Hostgroups and Activation Keys

As a good practice it is recommended to organize all OpenShift hosts into so called host collections so that we can reuse them for the advanced installation method later on. To archive this, Red Hat Satellite 6.1 can assign newly created hosts into host collections 'via activation keys'. For a detailed description of this process and it's configuration refer to [sat6-soe] Step 7: Automate your provisioning.

Configuring OpenShift3 host prerequisites

To configure all OpenShift3 prerequisites during provisioning of a new (or modified) host, we need to run Puppet modules via Red Hat Satellite 6.1.

We will use Red Hat Satellite 6.1 to prepare the docker storage parameters on each OpenShift node, we will use the pattern described in 'Configuring Docker Storage Option A)' [13: https://access.redhat.com/documentation/en/openshift-enterprise/version-3.0/openshift-enterprise-30-administrator-guide/chapter-1-installing#configuring-docker-storage].

Removing NetworkManager as required by [14: https://access.redhat.com/documentation/en/openshift-enterprise/version-3.0/openshift-enterprise-30-administrator-guide/chapter-1-installing#host-preparation] will also be done by a Puppet module.

As OpenShift will use it's own internal docker registry, we need to configure docker accordingly. An 'insecure registry' option [15: https://access.redhat.com/documentation/en/openshift-enterprise/version-3.0/openshift-enterprise-30-administrator-guide/chapter-1-installing#software-prerequisites] must be added to docker's configuration. Again, this will be done using a third Puppet module.

After installing OpenShift, you can choose to secure the integrated Docker registry, which involves adjusting the 'insecure registry' option accordingly.

Firewall prerequisites

OpenShift components provide their services via a shared network. To ensure communication between all components there must either be no firewalls between the hosts or firewall settings reflect the settings of the corresponding host firewalls.

Please see the OpenShift Enterprise 3 documentation for further details.

Deploying ACME Corp DEV environment

After planning and configuring your OpenShift Enterprise 3 environment within Red Hat Satellite 6.1, you need to create hosts, and install OpenShift Enterprise 3 using the advanced installation method.

Generating an Ansible inventory out of Red Hat Satellite 6.1

The OpenShift advanced installation method is based on Ansible, which needs an inventory file to work. This inventory contains a set of default parameters for the installation procedure and a classified set of hosts to deploy onto.

The classification set of hosts is mapped from Red Hat Satellite 6.1's by an Ansible Dynamic Inventory Source [16: https://docs.ansible.com/ansible/intro_dynamic_inventory.html]. A script that acts as a Dynamic Inventory Source will read the current host collections from Red Hat Satellite 6.1 and generate an Ansible Inventory.

The following mapping from Red Hat Satellite 6.1 host collections to Ansible groups is recommended:

Table 6. Mapping: hostgroups to groups

Red Hat Satellite 6.1 hostgroup	Ansible Inventory Group	Comment
OSE3 masters	masters	all master hosts
OSE3 nodes	nodes	all nodes
OSE3 etcd	etcd	all etcd nodes

Building an OpenShift Enterprise 3 Installer Container

We recommend that the 'advanced installation method' of OpenShift Enterprise 3 will be used to deploy and configure your environment.

To enable this in a very portable and standalone way, it is recommended to generate a docker container image containing all the required files:

- Ansible itself
- the ansible playbooks
- the dynamic inventory script
- required SSH keys to access remote hosts

Separating this software stack into a container will help keeping host systems clean: EPEL7 (required for Ansible) must not be installed and enable on any host, as it is separated in the container. The 'installer container' constitutes a single entity that could be version controlled and distributed easily.

Using 'ose3-installer' container image to do an advanced installation

Operational Considerations on OpenShift Enterprise 3

Managing OpenShift components

Management of OpenShift Enterprise 3 components should be performed by [IT-Ops-Plt-Ops]. This role is capable of updating software components (via updated Red Hat Satellite 6.1 Content Views) and configurations.

The OpenShift Enterprise 3 master components are manged via the command line interface, this includes the registry and router services.

While maintaining a node within the OpenShift Enterprise 3 cluster is a standard operation [17: https://access.redhat.com/documentation/en/openshift-enterprise/version-3.0/openshift-enterprise-30-administrator-guide#managing-nodes], you need to keep in mind, that NFS exports on the central storage system need to be updated if you add a new node: exports must match the new list of nodes.

Using 'ose3-installer' container image to do an advanced installation

During the design of ACME Corp's OpenShift Enterprise 3 platform a [ose3-install-container] image has been created. It will be used to perform the advanced installation method. Given a current Ansible inventory and a SSH public key, this container image could be generated to deploy to the most current set of hosts.

Backup and Restore

The following chapters will give a short overview of backup requirements and restore procedures for most of the OpenShift Enterprise 3 platform components. How to backup and restore the central NFS server component is not covered and highly dependent on the NFS solution deployed to implement that component.

Within the following chapters will contemplate 'components', this will include the software packages implementing the component and also configuration data of that component.

All backup should be performed by a 'Platform Operator'. Restore procedures should be executed by a 'Platform Operator', only reading and restoring of certificate data should be restricted even more.

integrated Certification Authority and Certificates

`/etc/openshift/master/` contains all configuration data of the OpenShift Enterprise integrated Certification Authority (CA). These files should be backed up in a secure location.

OpenShift Enterprise Master Components

Most of the configuration of the OpenShift Enterprise master components live within `etcd`. Just a few files are written to disk and need to be backed up separately.

Beside the configuration data living is `etcd`, no configuration files are altered while any master component is running, so it is save to backup the files without stopping any OpenShift Enterprise master services.

The following files need to be backed up:

Table 7. OpenShift Enterprise Master configuration file locations

directory	comments
`/etc/openshift/master/`	this contains all files related to CA and it's certificates, master, registry, router
`/etc/openshift/master/generated-configs/`	contains all configuration data generated during execution of the advanced installation method
`/var/lib/openshift/openshift.local.etcd/`	contains data of the integrated `etcd`, to be backed up by `etcdctl backup`
`/etc/sysconfig/openshift`	configuration of the openshift master service itself

Master integrated etcd Backup

OpenShift Enterprise 3 runs an 'etcd' as an integrated component within the openshift-master service. To back up 'etcd' data you need to stop 'openshift-master' service, back up `/var/lib/openshift/openshift.local.etcd/`, and start `openshift-master` again.

If 'openshift-master' service is stopped, neither to web console will give consistent results nor `oc` could be used to send commands to OpenShift Enterprise 3.

To do a backup of integrated 'etcd' data you must install the `etcd` RPM but must not start etcd systemd service in any way. Use `etcdctl backup --data-dir=/var/lib/openshift/openshift.local.etcd/` to backup data from the integrated 'etcd' component.

OpenShift Enterprise Nodes

While the 'openshift-master' component holds all configuration items 'openshift-node' should be considered data-less. All configuration items on OpenShift Enterprise 3 nodes are generated by the advanced installation method and we encourage an operational model of 'adding nodes' rather than 'restoring nodes'.

If an 'openshift-node' fails, and is in an state were a restore procedure would normally be executed to regain an operational state, the addition of a new 'openshift-node' and removal of the failed node is recommended. This will ensure that all configuration items are in sync with the infrastructure.

OpenShift Enterprise integrated Registry

The OpenShift integrated registry does contain any container image that is generated (or built) by OpenShift. All the applications source code, from which the container images were build, must be available via git. Either on a company's git server or on one that is hosted on the cloud. Therefor any image within the OpenShift integrated registry can be rebuilt from source.

Regardless of this fact, it is recommended to back up the registry, specifically for the reason that not all old versions of a container image can be rebuilt. A rollback of an deployment will not be possible if the old container image versions are not available.

To back up the registry two things can (and should) be done:

1. scale down the registry to zero replicas
2. back up the file system exported via NFS to the registry's pods

The first step should be taken to shut down all registry's pods, so that all transfers of images from and to the registry could finish first. In any way the registry implements a write forward pattern and does not overwrite existing data on disk in a way that would result in unsafe backups.

The second step will ensure a backup of the data on disk. We recommend backing up the partition that is backing the NFS export which is used by the registry's pods.

During a restore procedure this will lead to a situation where potentially unused data will be on that partition. This data will not be cleaned up by the registry.

OpenShift Enterprise Router Service

Any router within an OpenShift Enterprise 3 cluster is generated from a configuration, it does not keep and data that need to be backed up. We recommend that you establish operational procedures to recover all routers of your infrastructure automatically.

Creating, modifying and deleting a router (incl highly available routers) is described in [ose3-admin].

Service Accounts and Secrets

'ServiceAccount' [18: https://access.redhat.com/documentation/en/openshift-enterprise/version-3.0/openshift-enterprise-30-developer-guide/chapter-3-service-accounts] and 'Secrets' [19: https://access.redhat.com/documentation/en/openshift-enterprise/version-3.0/openshift-enterprise-30-developer-guide/chapter-10-secrets] objects are stored inside of etcd. They are covered by a [master components] backup.

Zero downtime upgrades

This chapter will give an overview how to perform any upgrade within the full software stack driving and application:

- the Standard Operating Environment
- OpenShift Enterprise 3 platform
- OpenShift services within the platform
- applications deployed ontop of the platform

Your applications should implement a 'rolling' or 'recreate' deployment strategy [20: https://access.redhat.com/documentation/en/openshift-enterprise/version-3.0/openshift-enterprise-30-developer-guide/chapter-7-deployments#strategies], so that a node (that is scheduled to be updated) could easily evacuated and your application is started on other nodes available in the cluster.

OpenShift Enterprise 3 services like the registry or routers are OpenShift applications by themselves: a migration of these services to other nodes and an upgrade of the corresponding pods is likewise easy. You must not take any special actions to be prepared for a zero downtime upgrade.

You need to deploy highly available routers if you would like to implement zero downtime updates.

OpenShift's nodes and masters within a major version of OpenShift Enterprise 3 are forward and backward compatible, unless otherwise notes. Upgrading a cluster should go smoothly. However, you should not run mismatched versions longer than necessary to upgrade the entire cluster.

Prerequisites and procedures for upgrading OpenShift have been described within the Administrators Guide [21: https://access.redhat.com/documentation/en/openshift-enterprise/version-3.0/openshift-enterprise-30-administrator-guide/chapter-2-upgrading-openshift].

Using an automated deployment method as described in Deploying OpenShift Enterprise 3 is another important step to implement a zero downtime upgrade procedure. Using this deployment method will ensure that updates of all components are deployed efficiently on the Standard Operating Environment level.

Further information, including a detailed technical implementation of a zero downtime upgrade procedure is provided by [ose3-zero-downtime-upgrades].

Capacity Management

Managing the capacity of a platform is a constant **plan-do-check-act** [22: https://en.wikipedia.org/wiki/PDCA] cycle.

'Platform Operator' will **plan** the future resource requirements with 'Application Operator'.

Adding or upgrading nodes (**do**) will be performed by 'Platform Operator' in close cooperation with 'Infrastructure Operator'. Setting limits and quotas by 'Platform Operator' is in addition to that.

Monitoring of resource consumption of platform and infrastructure level (**check**), with sumarizing reporting to 'Platform Operator' is an ongoing effort, and the base to **act** upon the current resource usage.

Therefore Capacity Management within OpenShift Enterprise 3 is threefold:

- providing enough CPU and Memory resources to you platform
- managing storage to be provisioned to applications
- managing quotas and limits

Providing CPU and Memory

Providing enough CPU and Memory resources to applications could be achived by scaling OpenShift nodes. Most often this means scaling them up.

Two scenarios should be considered:

- scaling in a physical environment
- scaling in a virtual environment

As ACME Corp has implemented all OpenShift components based on virtual machines, scaling CPU and memory up is easy: one could either resize the VM within Red Hat Enterprise Virtualization 3.5 or a new node could be added to the cluster that is using 'a bigger flavor' for instance creating within Red Hat Enterprise Linux OpenStack Platform 7.

Scaling in a physical world should also be achived by adding new larger nodes to the OpenShift cluster.

Any of these maintenance activities should be carried out by the 'Infrastructure Operator' role.

Application Storage provisioning

As applications that are hosted on the OpenShift Enterprise 3 platform require storage (e.g. for database files, or images) an Infrastructure Operator needs to provide an backing component for these requirements. With OpenShift Enterprise 3 the only supported protocol for storage provisioning is NFS.

An Infrastructure Operator will provide a set of NFS exports, according to Providing Storage to Pods. These NFS exports will be provisioned to OpenShift Enterprise 3 via `PersistentVolume` configuration items, with enforced quotas if desired.

As provisioning of storage is a mass configuration, mount points for partitions on NFS servers, exports, and persistent volumes should follow a generic naming scheme and be configured automatically via a script.

Quotas and Limits

While Limits implement soft boundaries on resources, Quotas implement hard boundaries, which cannot be broken by eg. a container.

Quota and limit maintenance should be performed by a 'Platform Operator' as this role needs to maintain a consolidated overview of consumed resources.

Limits on Projects

Limits may be applied to any object in a project that is consuming cpu or memory, therefor a limit can be applied to pods and to containers in this pod. A Limit may have a minimum, default and maximum value assigned to it. As an IT-Ops, Application Operator you are responsible for setting the limits via the OpenShift Enterprise 3 client [23: https://access.redhat.com/documentation/en/openshift-enterprise/version-3.0/openshift-enterprise-30-developer-guide/chapter-12-resource-limits] application.

Quotas on Resources

After a quota is created in a project, the project restricts the ability to create any new resources that may violate a quota constraint until it has calculated updated usage statistics. If a modification to a project would exceed a quota usage limit, the action is denied by the server.

Quotas on Persistent Volumes

To enforce quotas on persistent volumes it is recommended to use disk partitions as NFS exports. As persistent volume claim will be mapped to a persistent volume of the same size or bigger. This persistent volume will than be backed by a NFS exports of the same size (assuming that the pool of that sized exports is not empty (or bound to other pods)).

An Infrastructure Operator need to ensure that NFS exports are configured in accordance with this recommendation.

Appendix A: Vision History

Table 8. OSE3 Ref Arch Version History

Version	Comment	Release date	Released by
0.8.0	Initial Release for SysEng peer review	2015-Sept-03	Christoph Görn
0.9.0	Initial Release for OpenShift stakeholder review	2015-Sept-11	Christoph Görn
1.0.1	Initial Public Release	2015-Sept-24	Christoph Görn
Document generated by Asciidoctor PDF			

Appendix B: Contributors

Aaron Weitekamp, content review

Bibliography

- [ose3-admin] Red Hat OpenShift Documentation Team. OpenShift Enterprise 3.0 Administration. Red Hat. 2015.
- [sat6-soe] Dirk Herrmann, 10 Steps to build a Standard Operating Environment, Red Hat. 2015.
- [rhsat61] Red Hat Satellite Documentation Team. Red Hat Satellite 6.1 User Guide. Red Hat. 2015.
- [rhev35] Red Hat Enterprise Virtualization Documentation Team, Red Hat Enterprise Virtualization 3.5, Red Hat, 2015.
- [rhelosp7] OpenStack Documentation Team, Red Hat Enterprise Linux OpenStack Platform 7, Red Hat, 2015.
- [rhel7-idm] Ella Deon Ballard, Tomáš Čapek, Aneta Petrová. Linux Domain Identity, Authentication, and Policy Guide. Red Hat. 2015.
- [ose3-ha-refimpl] Scott Collier, Deploying an OpenShift Enterprise 3 Distributed Architecture, Red Hat. 2015
- [ose3-zero-downtime-refimpl] N. N., Implementing a Zero Downtime upgrade procedure for OpenShift Enterprise 3, Red Hat, 2015
- [osp-aep] Jon Jozwiak, AEP on RHEL OSP, Red Hat, 2015