Red Hat Taste of Training

Ben Breard, RHCA, RHCDS, RHCE, RHCVA
Solutions Architect, Red Hat
bbreard@redhat.com

Thomas Cameron, RHCA, RHCDS, RHCE, RHCVA, RHCSS, RHCX
Chief Architect, Western US, Red Hat
thomas@redhat.com

PRESENTED BY RED HAT

LEARN. NETWORK.
EXPERIENCE OPEN SOURCE.

www.theredhatsummit.com

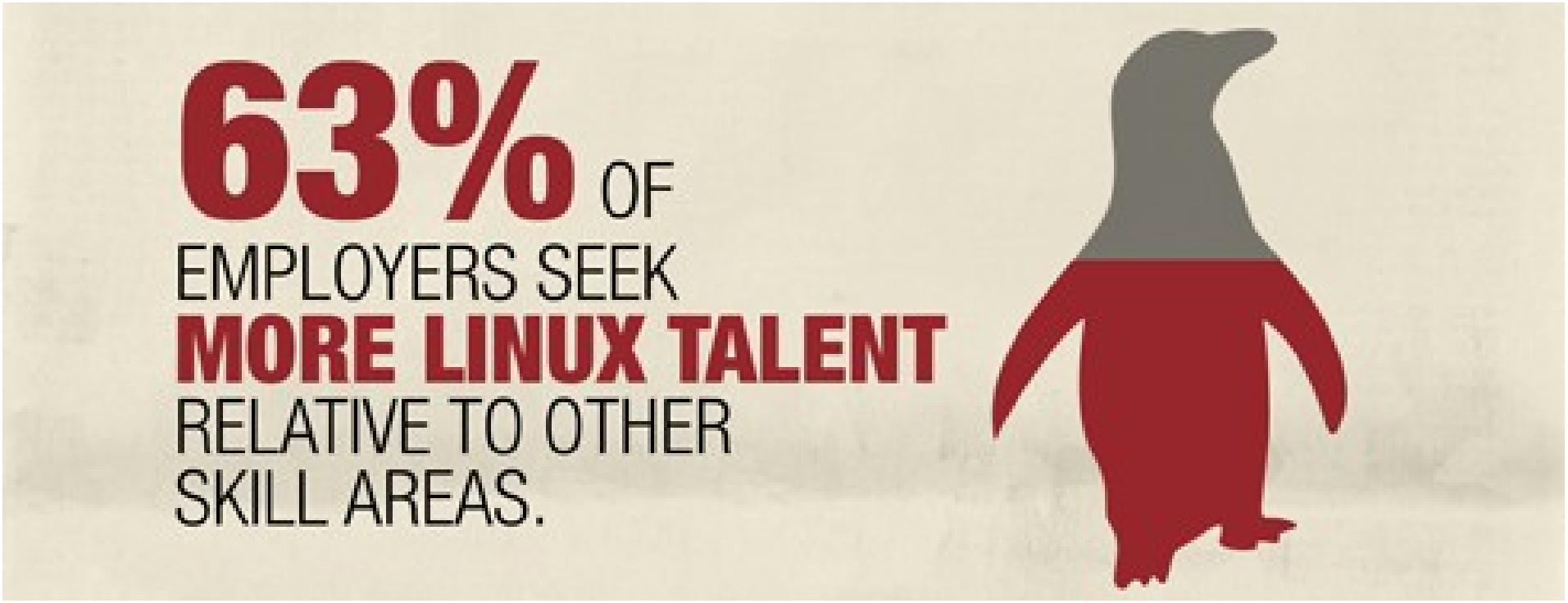
Agenda

- Red Hat Training Overview
- Performance Tuning with Tuned & Numad
- Building RPMs

  Break
- SELinux for Mere Mortals
- OpenShift Overview, Architecture & App Deployment

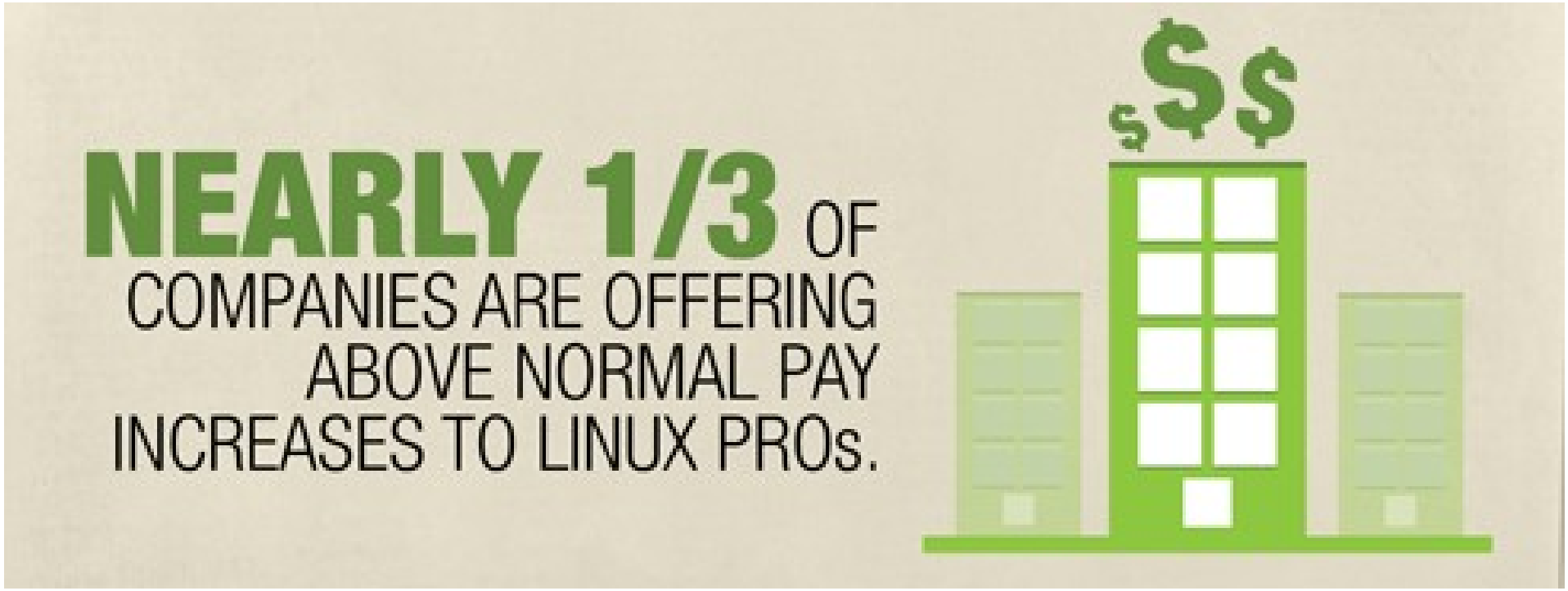
2012 Linux Jobs Report

2012 Linux Jobs Report



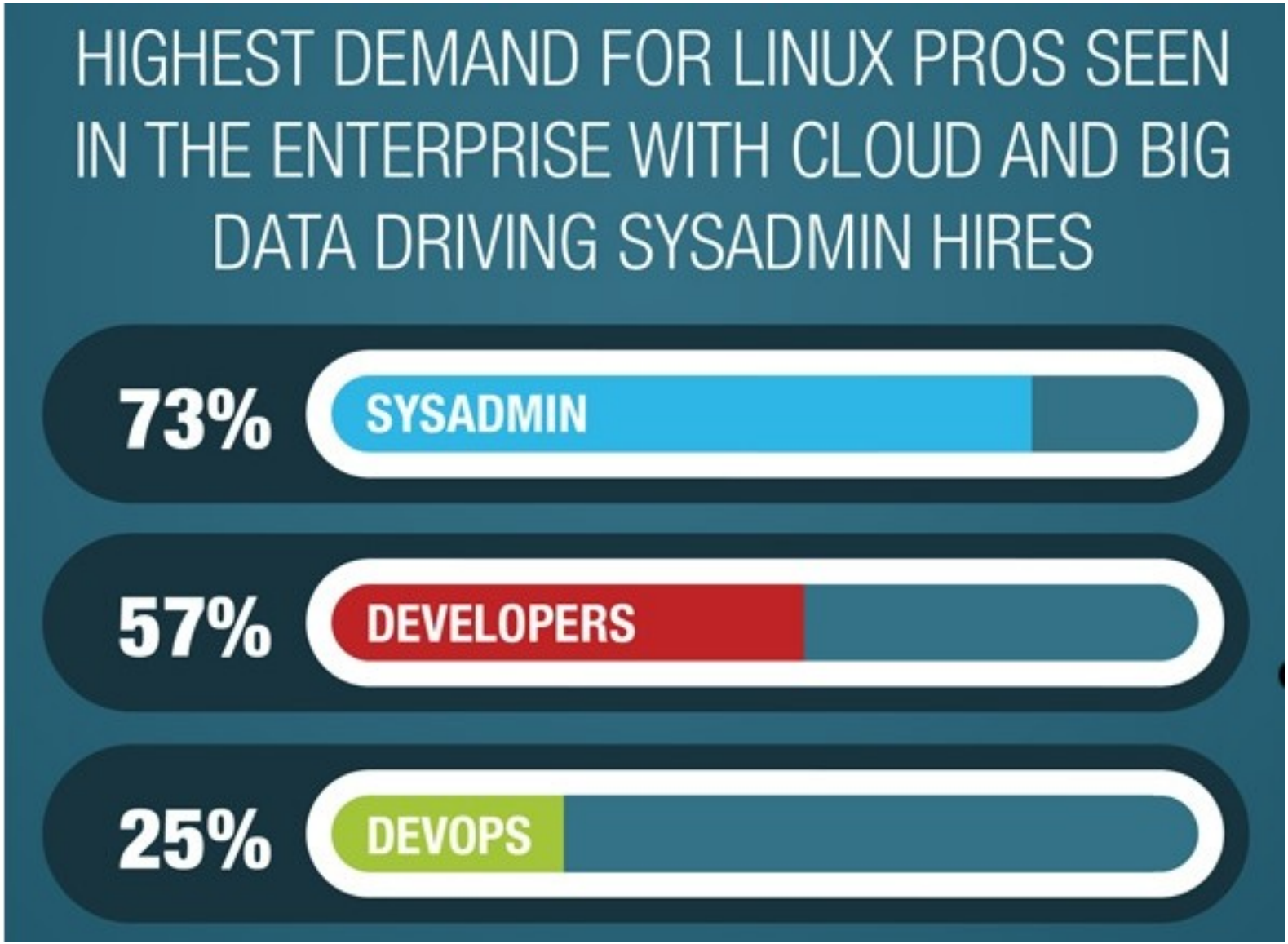
2012 Linux Jobs Report

2012 Linux Jobs Report



2013 Linux Jobs Report

2013 Linux Jobs Report

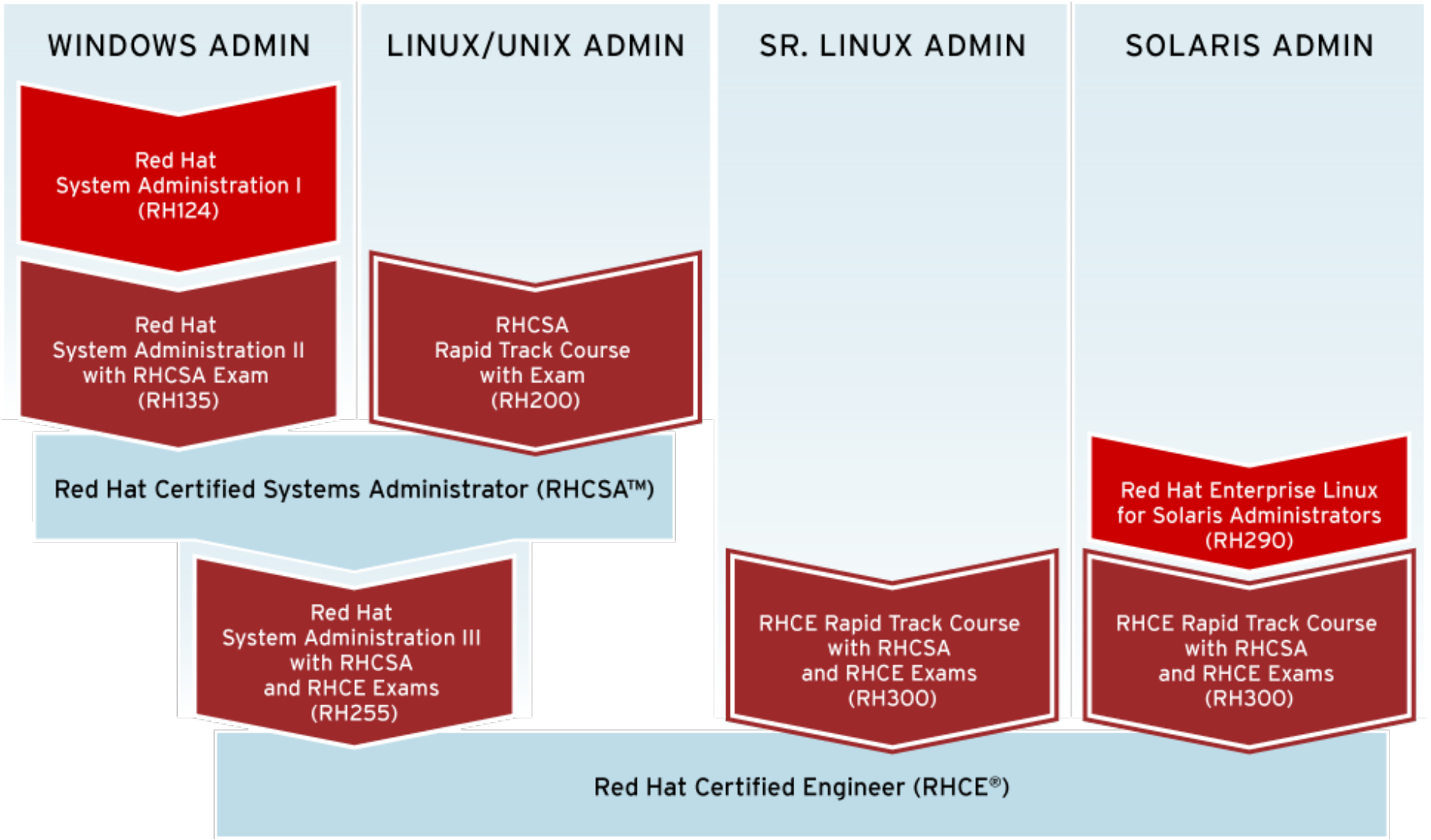


Red Hat Training

- Over thirty classes to fit area of interest and skill level.
- Flexible ways to consume
  - Instructor-led classroom
  - Virtual classroom
  - Online – self paced
  - On-site training
- Quality of content

Red Hat Enterprise Linux System Administration

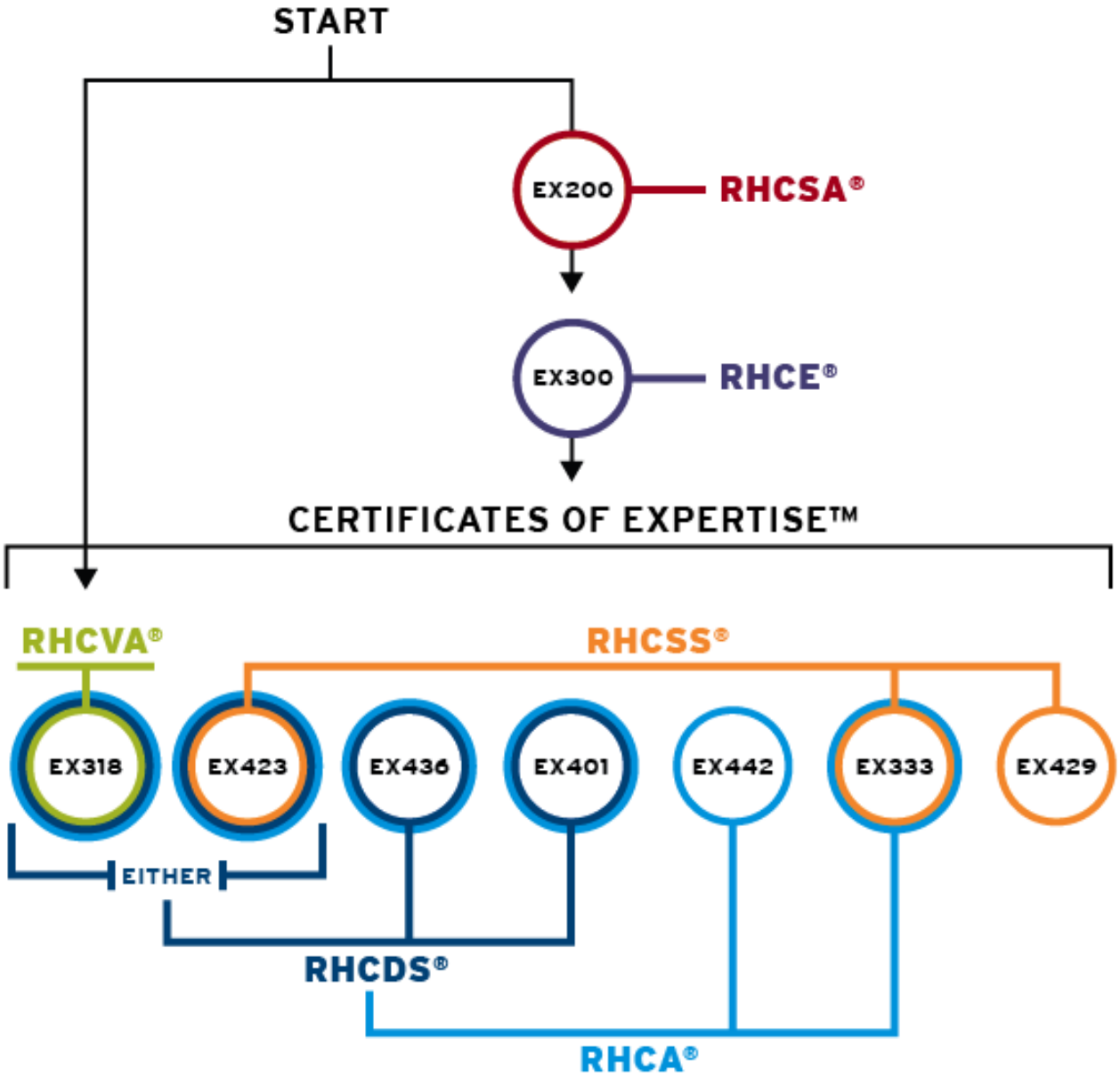
Core Training Paths



Red Hat Certifications

- Certifications are highly regarded
- All exams are performance based
  - Offered on Friday mornings and afternoons
  - Kiosk tests available in most major cities

Certification Path



Advanced System Administration Classes

- RH318 - Red Hat Enterprise Virtualization
- RH423 - Directory Services and Authentication
- RH436 - Clustering and Storage Management
- RH401 - Deployment and System Management
- RH442 - Performance Tuning
- RH333 - Enterprise Security
- RH429 - SELinux Policy Administration

PERFORMANCE TUNING WITH

TUNED & NUMAD

Pronounced “Tune Dee” & “Numa Dee”

Adapative Tuning Daemon

Nine default profiles

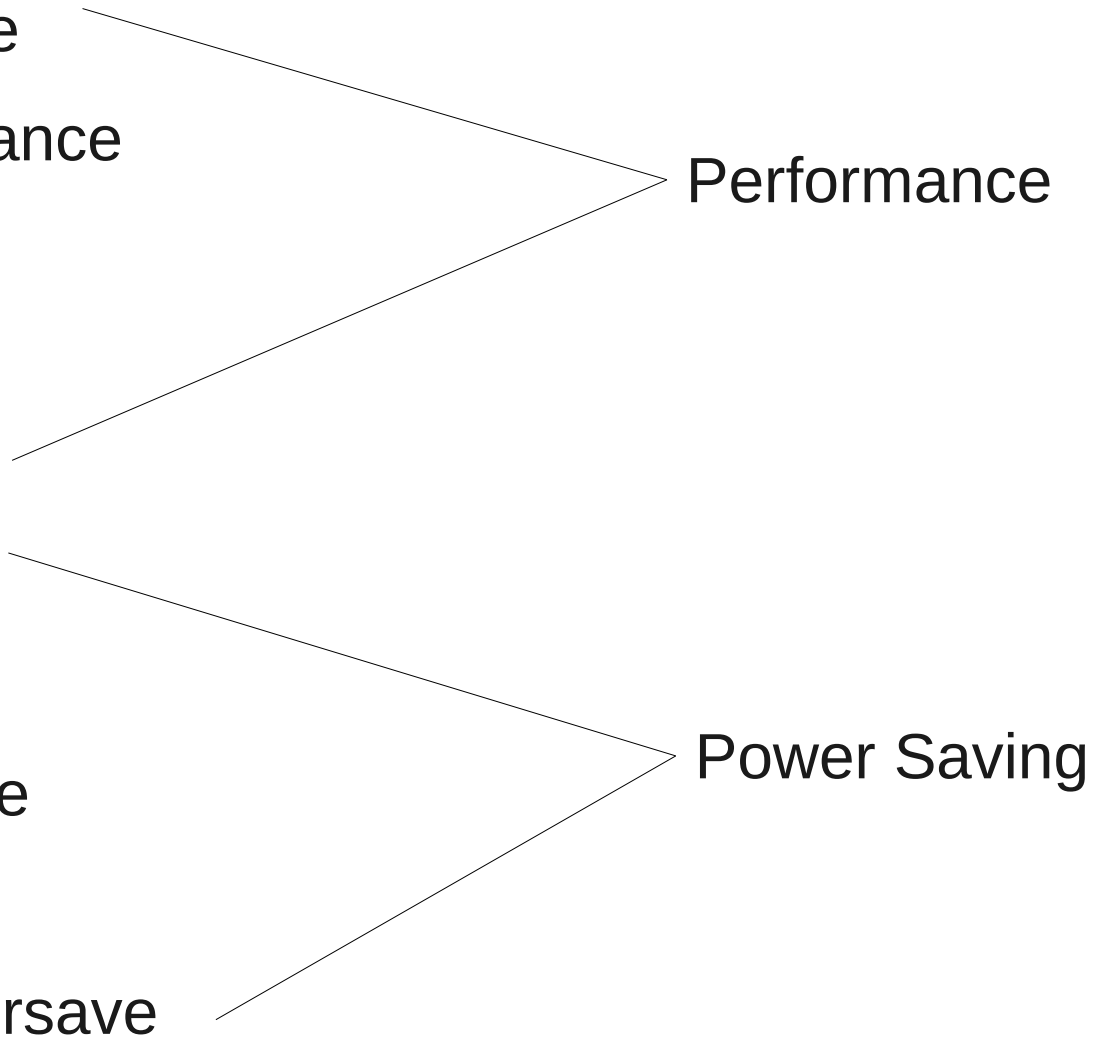
Switch profiles on-the-fly

Profiles

- default
- latency-performance
- throughput-performance
- enterprise-storage
- virtual-guest
- virtual-host
- spindown-disk
- desktop-powersave
- laptop-ac-powersave
- server-powersave
- laptop-battery-powersave

Profiles

- default
- latency-performance
- throughput-performance
- enterprise-storage
- virtual-guest
- virtual-host
- spindown-disk
- desktop-powersave
- laptop-ac-powersave
- server-powersave
- laptop-battery-powersave



RHEL 6 Profiles Comparison

Tunable	default	latency-performance	throughput-performance	enterprise-storage	virtual-host	virtual-guest
kernel.sched_min_granularity_ns	4ms	3ms	10ms	10ms	10ms	10ms
kernel.sched_wakeup_granularity_ns	4ms	3ms	15ms	15ms	15ms	15ms
vm.dirty_ratio	20% RAM		40%	40%	10%	40%
vm.dirty_background_ratio	10% RAM				5%	
vm.swappiness	60				10	30
I/O Scheduler (Elevator)	CFQ	deadline	deadline	deadline	deadline	deadline
Filesystem Barriers	On			Off	Off	Off
CPU Governor	ondemand	performance	performance	performance	performance	performance
Disk Read-ahead				4x	4x	4x

```
# tuned-adm list
# tuned-adm active
# tuned-adm profile <profile-name>
# tuned-adm off
```

Create your own profiles

/etc/tune-profiles/<*profile-name*>

tuned.conf
ktune.sysconfig
sysctl.ktune
ktune.sh

tuned.conf

Enable/Disable Power Management Plugins
CPU
Disk
Network

ktune.sysconfig

Enable/Disable ktune daemon
Configure disk elevators

sysctl.ktune

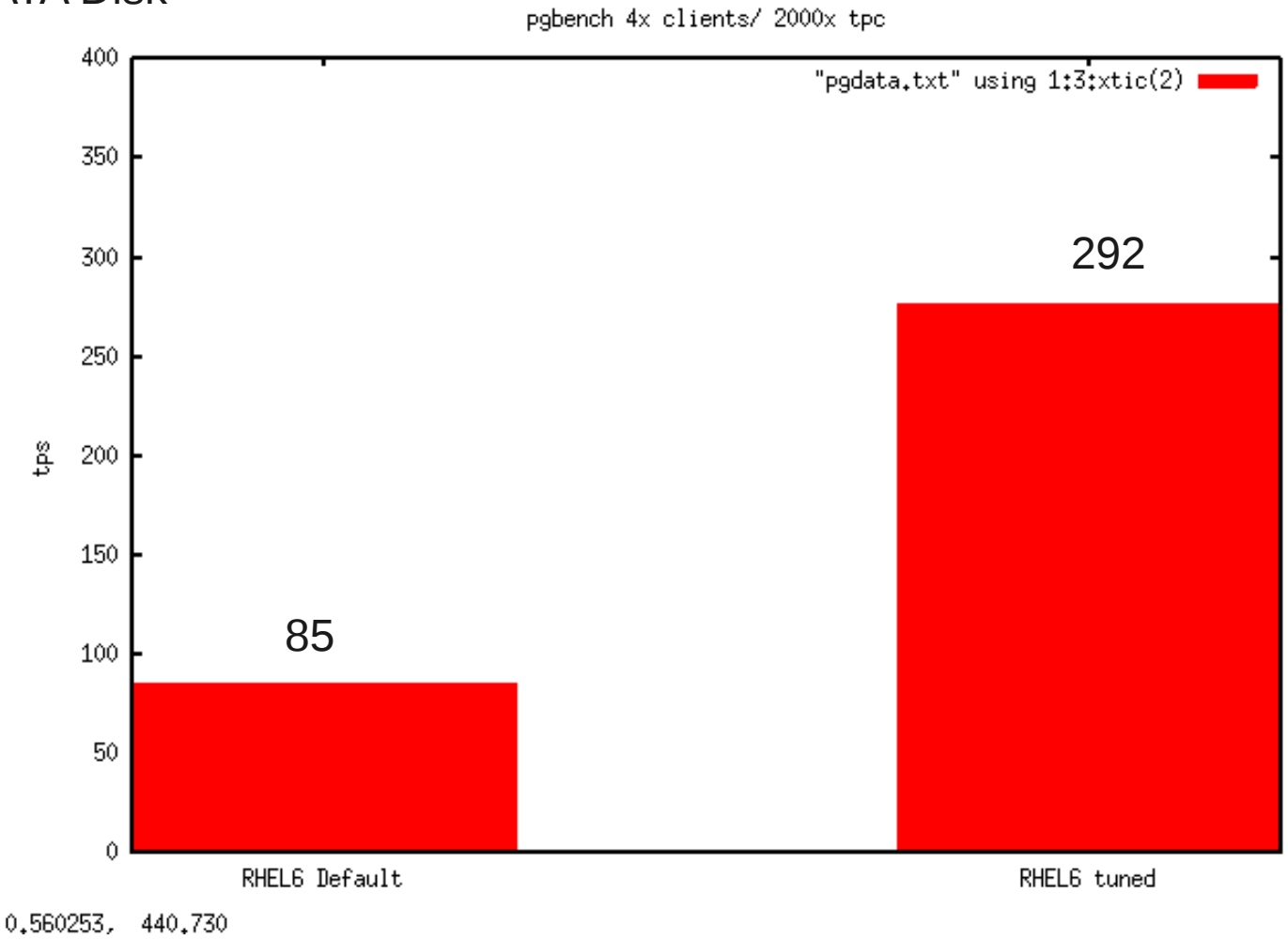
`sysctl` settings

ktune.sh

start() and stop()
/etc/tune-profiles/functions

Example 1

Intel Core 2 Duo T9400 2.53GHz
7200 rpm SATA Disk

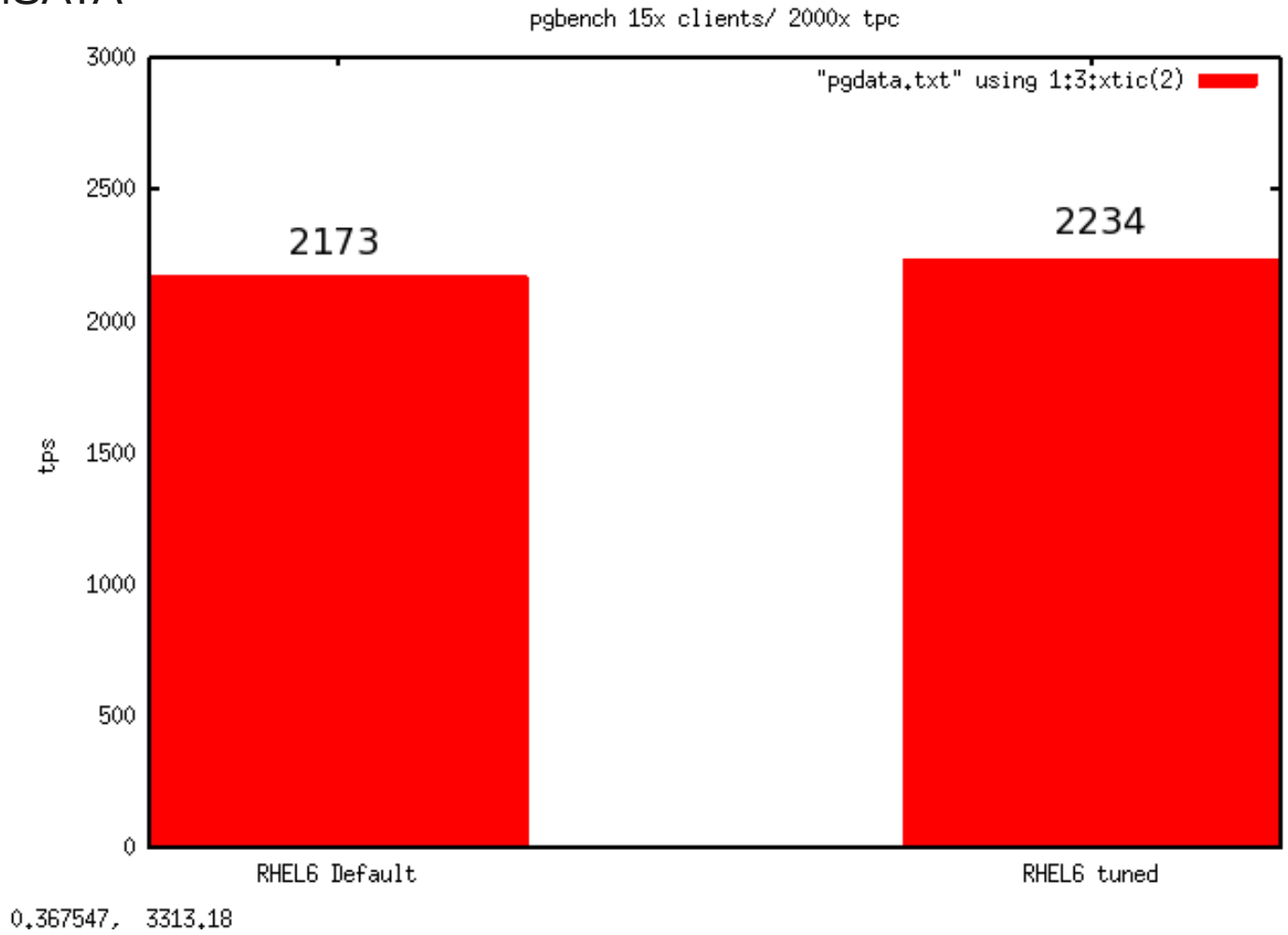


Example 1 - Results

- The disk is the bottle neck
- Write barriers and deadline provide the largest benefit
- Not all scenarios will yield a 244% increase.

Examples 2

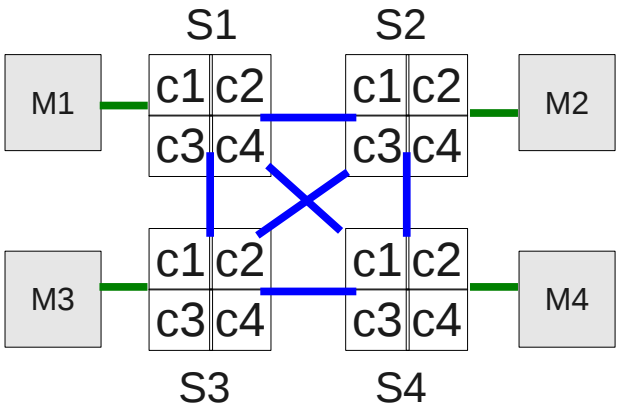
Intel Core i7-2620M CPU 2.70GHz
OCZ SSD MSATA

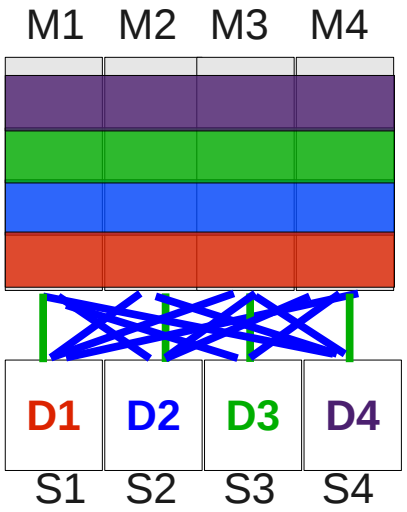


Examples 2

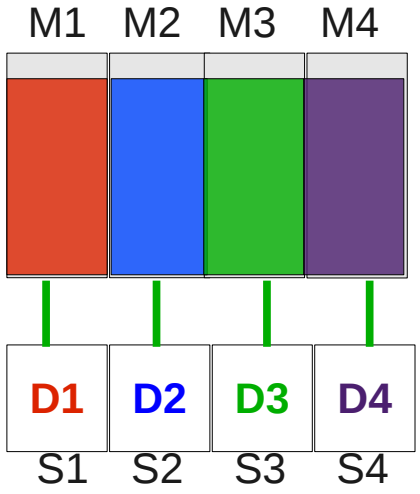
- Yielded 2.8% improvement.
- IO is not the bottle neck
- Proves that hardware, environment, workload, etc must be taken into consideration for tuning.

RHEL and NUMA (Non Uniform Memory Access)





No NUMA optimization



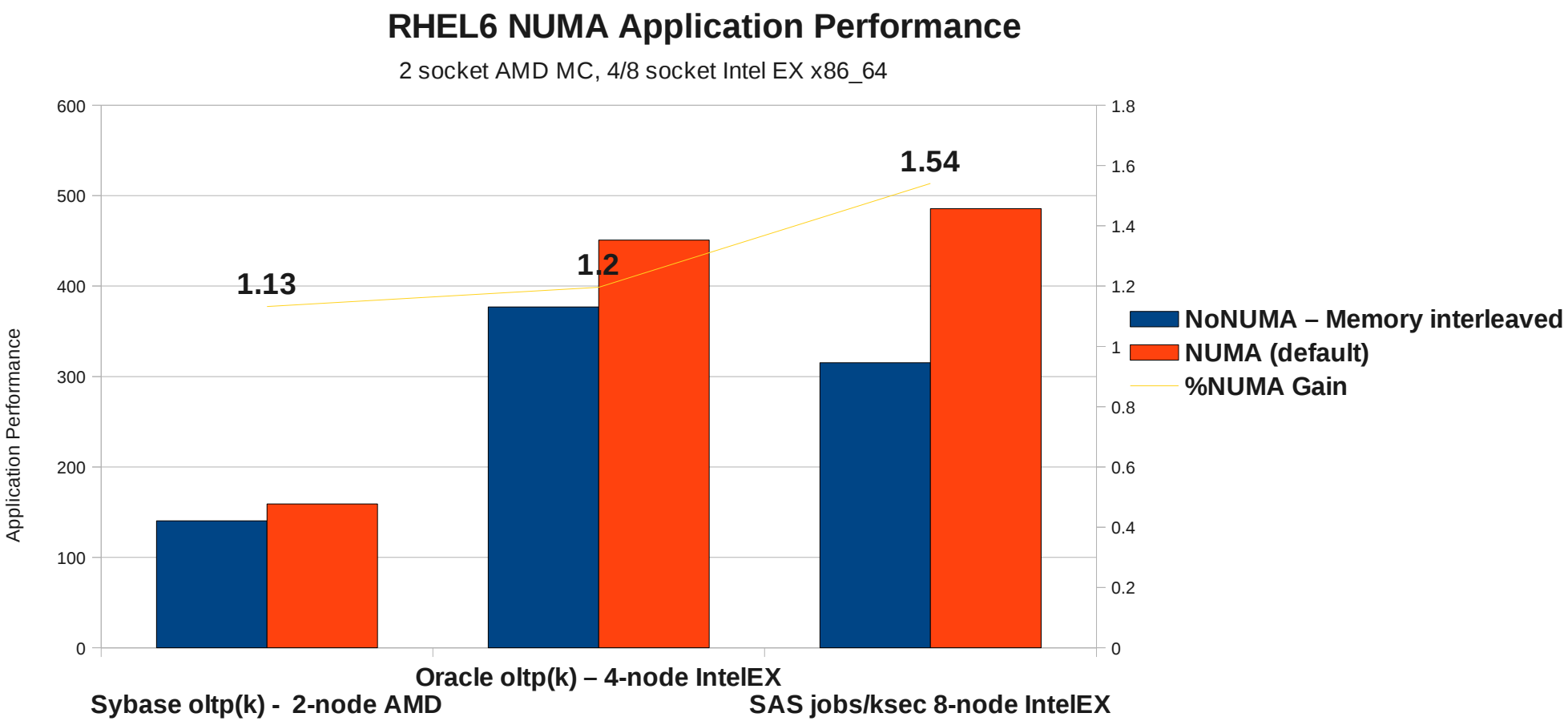
NUMA optimization

- Multi Socket – Multi core architecture
  - X86 memory controllers on-chip all AMD/Intel servers
  - RHEL 5 / 6 NUMA aware
  - Additional performance gains by enforcing NUMA placement

Finding NUMA layout

```
[root@perf30 ~]# numactl --hardware
available: 4 nodes (0-3)
node 0 cpus: 0 4 8 12 16 20 24 28 32 36 40 44 48 52 56 60
node 0 size: 32649 MB
node 0 free: 30868 MB
node 1 cpus: 1 5 9 13 17 21 25 29 33 37 41 45 49 53 57 61
node 1 size: 32768 MB
node 1 free: 29483 MB
node 2 cpus: 2 6 10 14 18 22 26 30 34 38 42 46 50 54 58 62
node 2 size: 32768 MB
node 2 free: 31082 MB
node 3 cpus: 3 7 11 15 19 23 27 31 35 39 43 47 51 55 59 63
node 3 size: 32768 MB
node 3 free: 31255 MB
node distances:
node   0   1   2   3
  0:  10  21  21  21
  1:  21  10  21  21
  2:  21  21  10  21
  3:  21  21  21  10
```

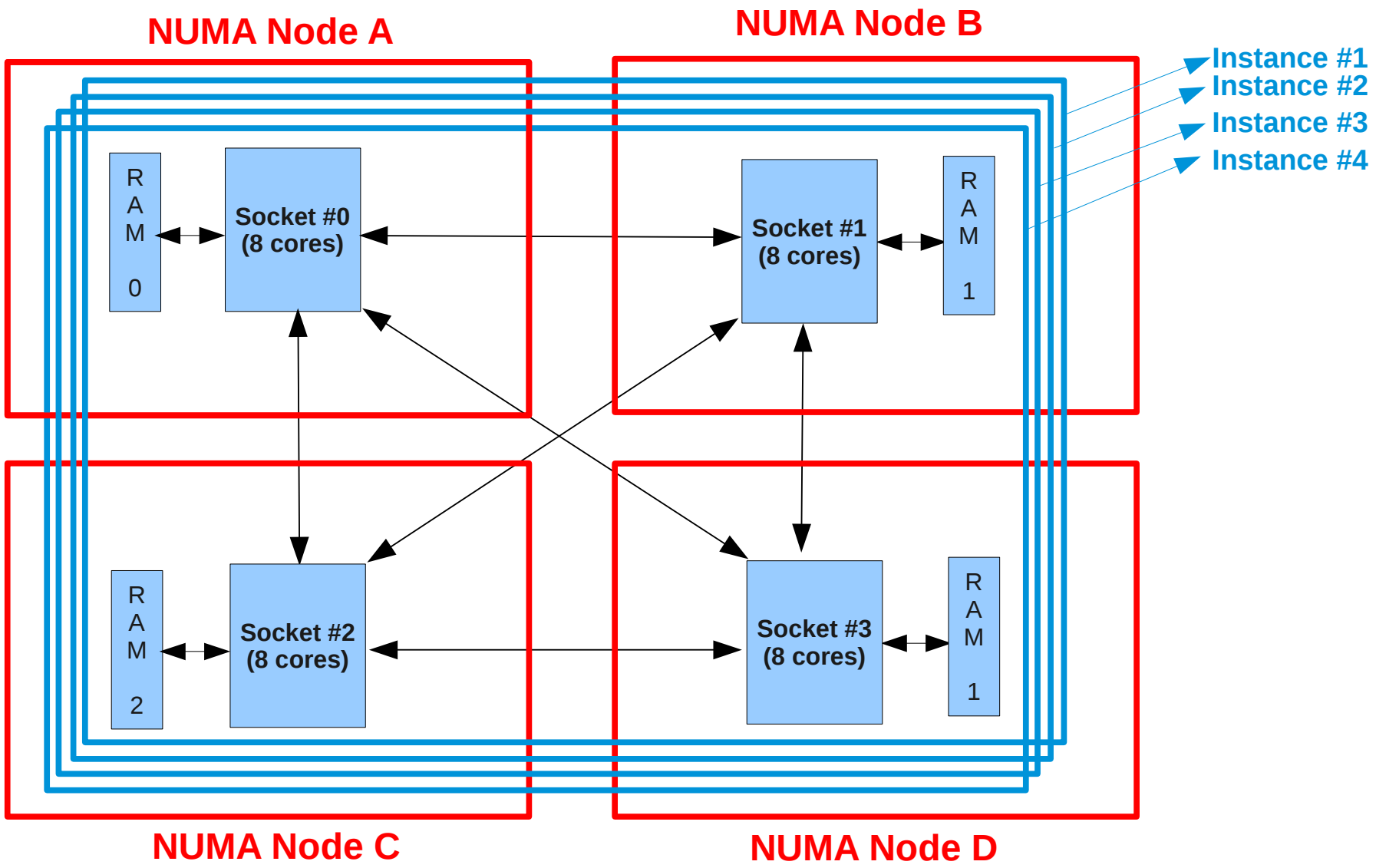
RHEL6 Differences w/NUMA Multi-node Databases



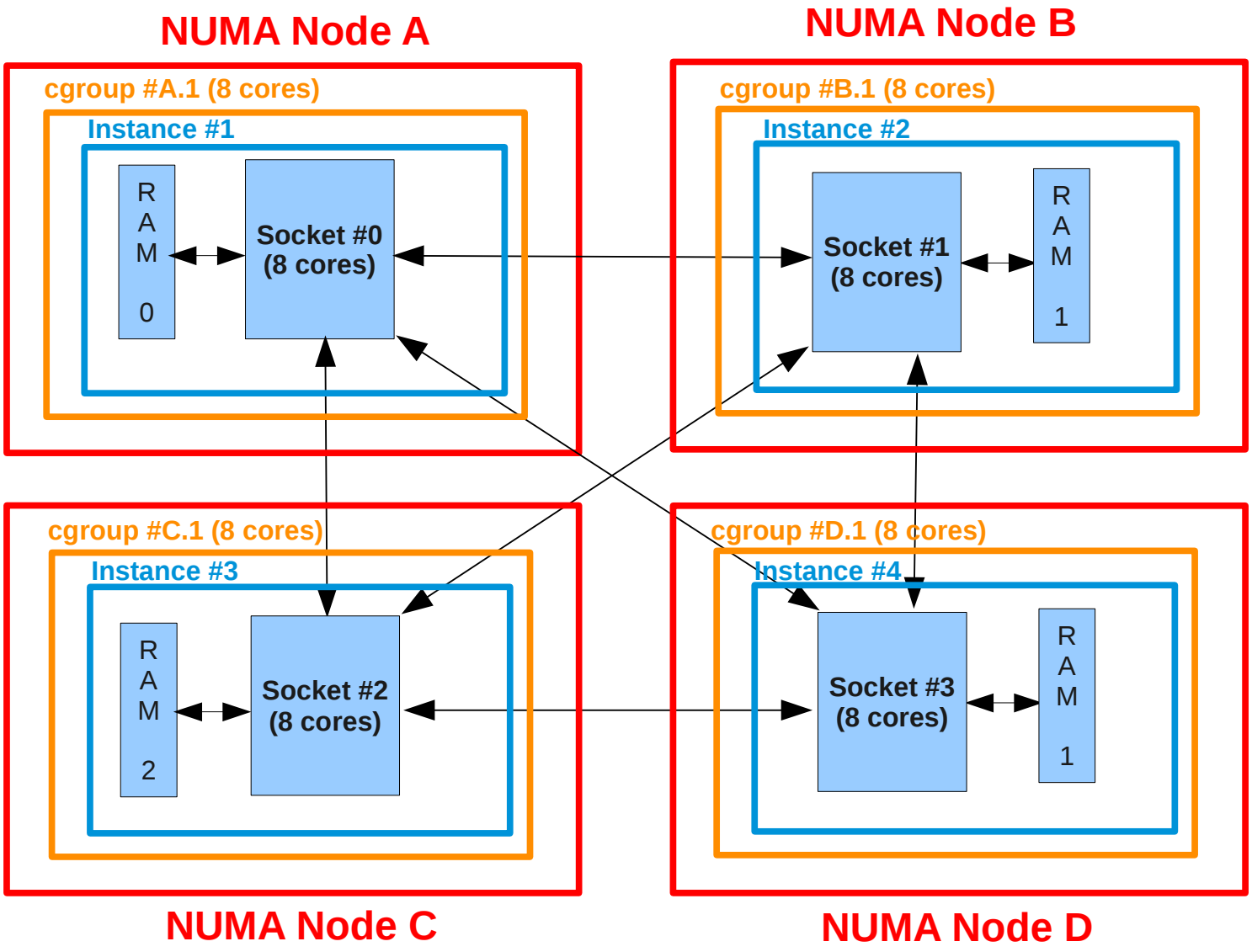
Numa – Latency

- Sample worstcase-NUMA-node relative latency:
  - Intel 4 socket / 4 node: 1.5x
  - AMD 4 socket / 8 node: 2.7x
  - 8 socket / 8 node: 2.8x
  - 32 node blade system: 5.5x

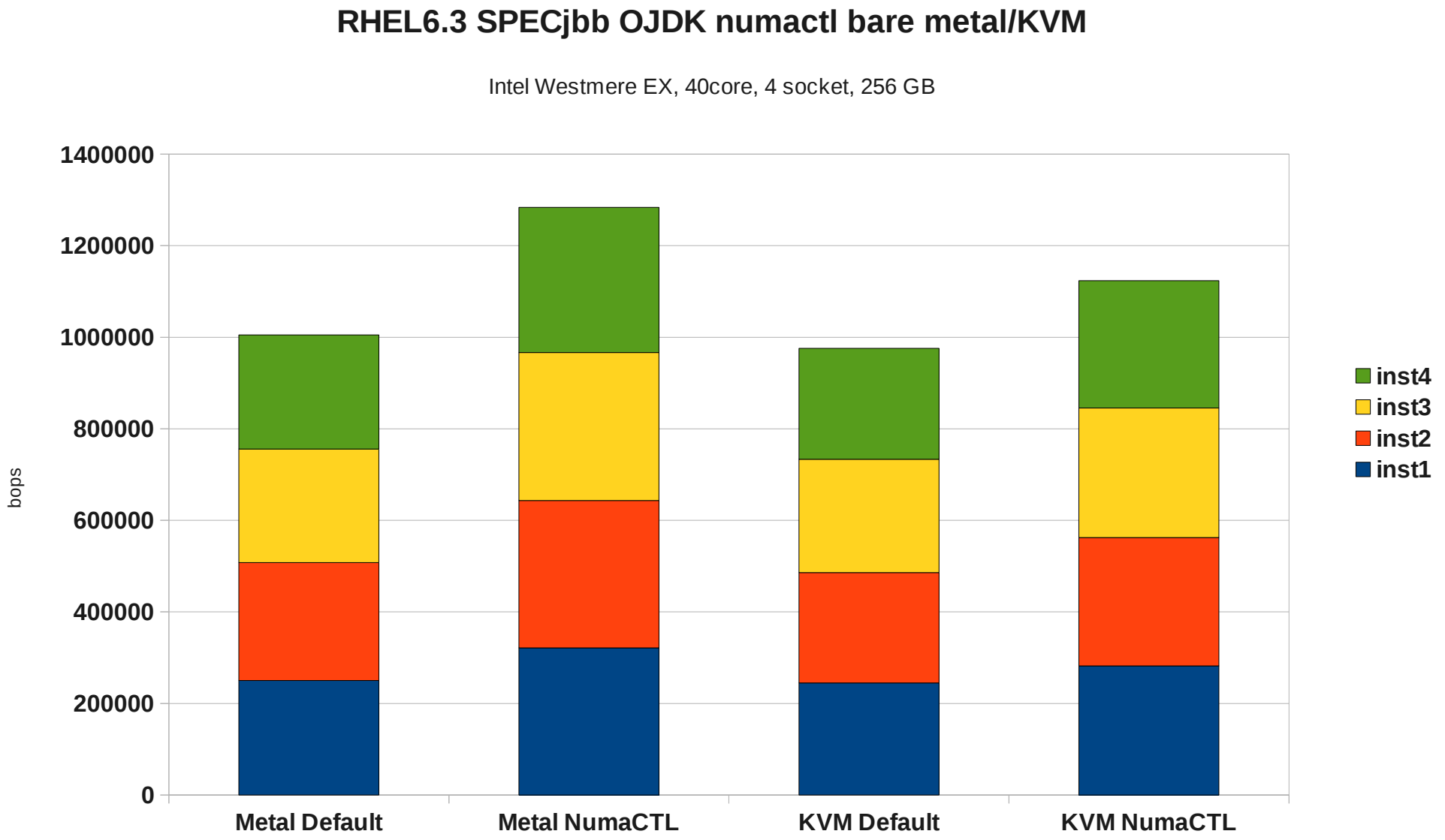
Default Scheduler: 4 Unpinned



NUMAD 4 Apps, pinned to the 4 sockets



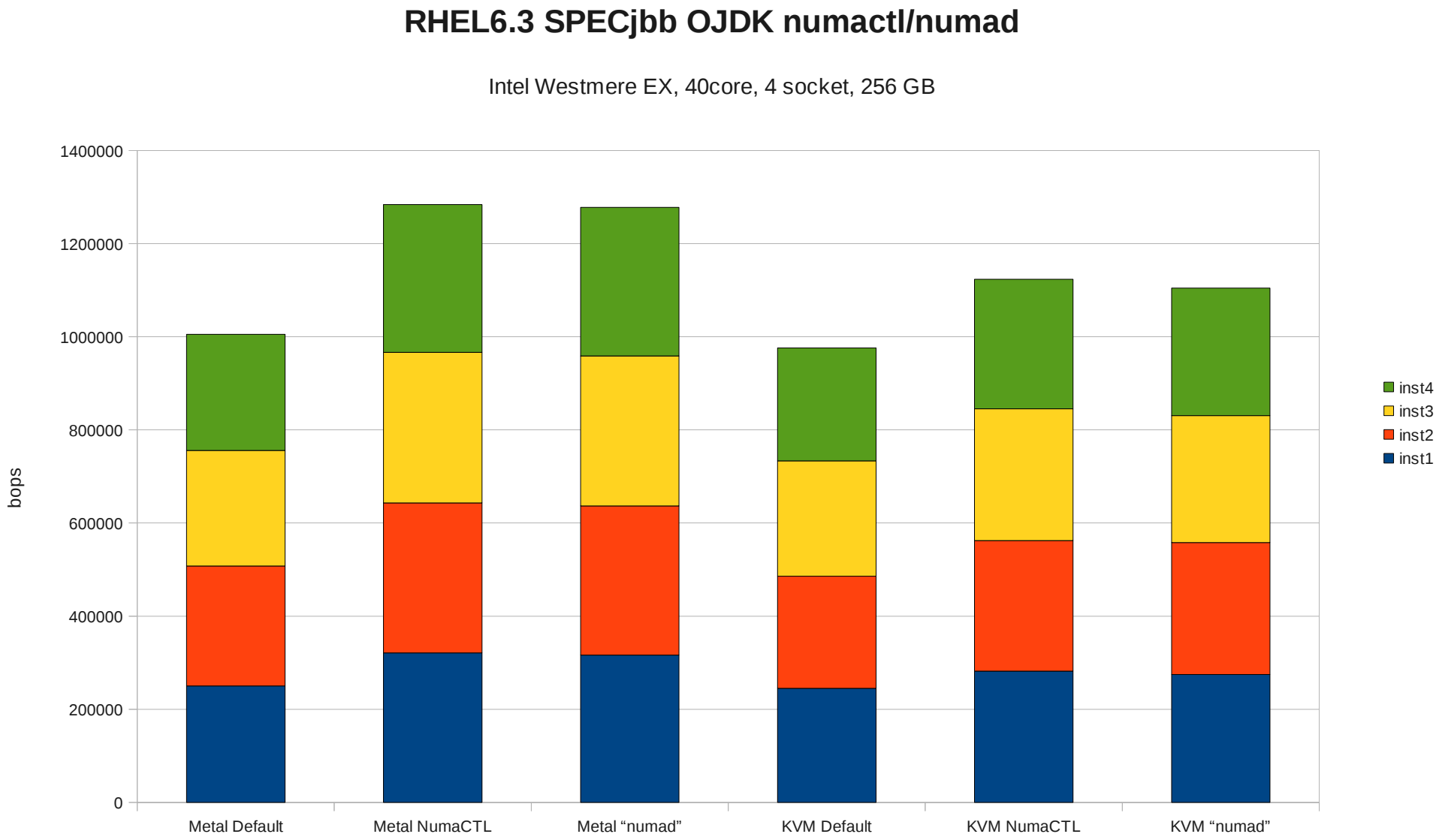
RHEL6 Differences w/NUMA Multi-instance OJDK Java Perf



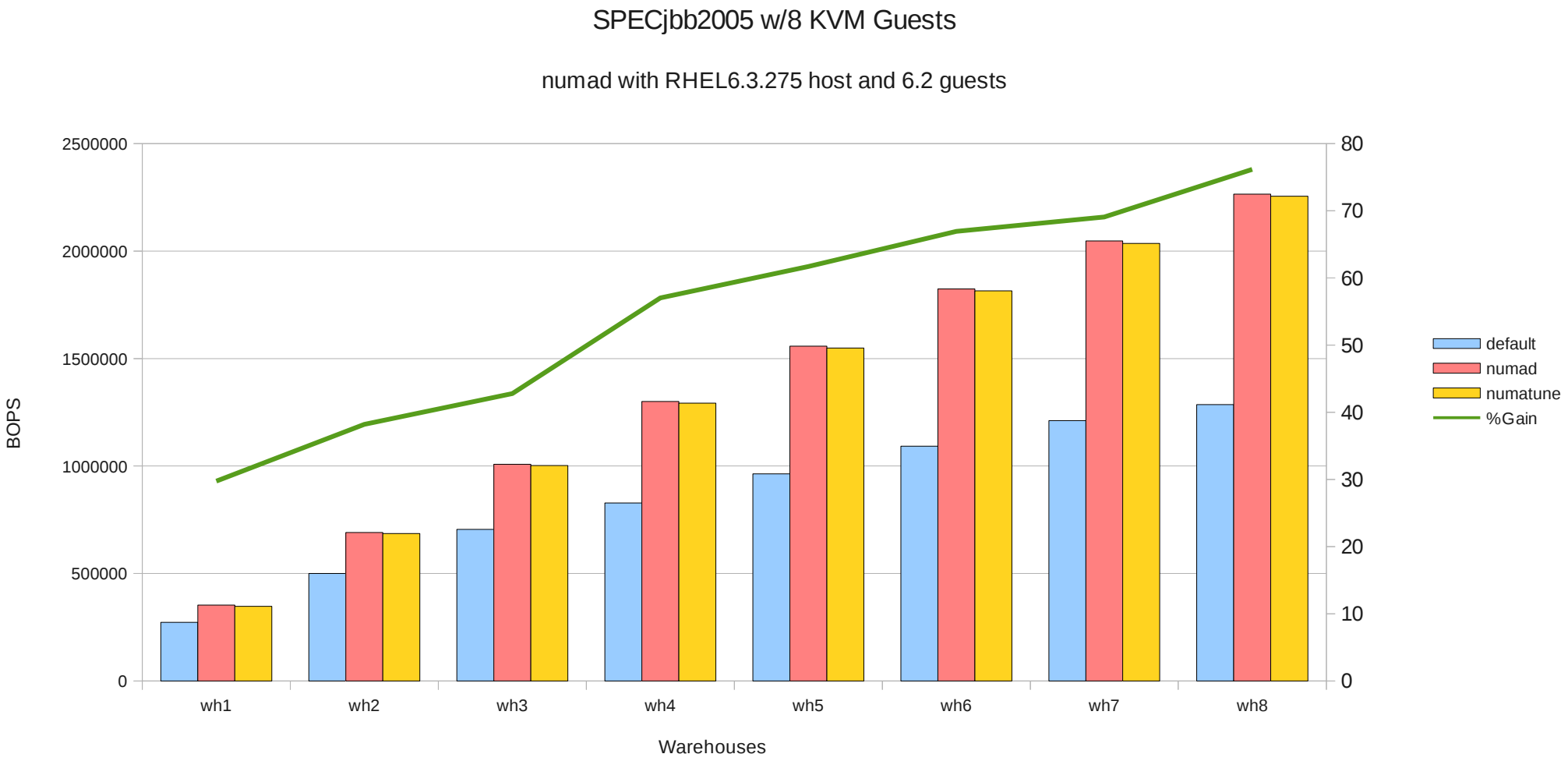
RHEL6.3 – NUMAD

- Aims to improve out of the box NUMA system performance for non-expert users
  - Optimizes best for multiple apps (metal/kvm) which fit within a numa-node
- A user space tool with 2 modes of operation
  - One shot pre-placement advice for the best initial process placement and resource affinity (libvirt usage).
  - Daemon mode where it continually monitors available system resources and significant consumers to allocate NUMA aligned resources for optimum performance.

RHEL6.3 SPECjbb opt w/ numad ~= numactl



KVM Multi-guest SPEC JBB – 8 node HP DL980



Example – RHEL Hypervisor without numad

File Edit View Search Terminal Help

```
[root@rhel-h2.rdu ~]$ numastat -n qemu

Per-node process memory usage (in MBs)
PID                         Node 0          Node 1           Total
---------------  --------------- --------------- ---------------
3568 (qemu-kvm)            583.24          577.30         1160.54
3807 (qemu-kvm)           5486.69         6716.62        12203.31
4488 (qemu-kvm)            659.03         1643.70         2302.73
7672 (qemu-kvm)            711.23         4984.16         5695.38
14250 (qemu-kvm)          2159.01         2019.75         4178.77
14823 (qemu-kvm)           484.36          695.58         1179.95
---------------  --------------- --------------- ---------------
Total                    10083.56        16637.11        26720.67

Per-node numastat info (in MBs):
                            Node 0          Node 1           Total
                  --------------- --------------- ---------------
Numa_Hit              21081785.76     18074538.80     39156324.56
Numa_Miss                 2268.10          134.90         2403.00
Numa_Foreign               134.90         2268.10         2403.00
Interleave_Hit              54.49           54.68          109.17
Local_Node            21081574.10     18074490.33     39156064.43
Other_Node                2479.76          183.36         2663.12
[root@rhel-h2.rdu ~]$
```

Example – RHEL Hypervisor without numad

File Edit View Search Terminal Help

Vms are split between nodes

```
[root@rhel-h2.rdu ~]$ numastat -n qemu

Per-node process memory usage (in MBs)
PID                         Node 0          Node 1           Total
---------------  --------------- --------------- ---------------
3568 (qemu-kvm)            583.24          577.30         1160.54
3807 (qemu-kvm)           5486.69         6716.62        12203.31
4488 (qemu-kvm)            659.03         1643.70         2302.73
7672 (qemu-kvm)            711.23         4984.16         5695.38
14250 (qemu-kvm)          2159.01         2019.75         4178.77
14823 (qemu-kvm)           484.36          695.58         1179.95
---------------  --------------- --------------- ---------------
Total                    10083.56        16637.11        26720.67

Per-node numastat info (in MBs):
                            Node 0          Node 1           Total
                   --------------- --------------- ---------------
Numa_Hit             21081785.76     18074538.80     39156324.56
Numa_Miss                2268.10          134.90         2403.00
Numa_Foreign              134.90         2268.10         2403.00
Interleave_Hit             54.49           54.68          109.17
Local_Node           21081574.10     18074490.33     39156064.43
Other_Node               2479.76          183.36         2663.12
[root@rhel-h2.rdu ~]$
```

Example – RHEL Hypervisor with numad

File Edit View Search Terminal Help

```
[root@rhel-h5 ~]# numastat -n qemu

Per-node process memory usage (in MBs)
PID                         Node 0          Node 1           Total
---------------  --------------- --------------- ---------------
10308 (qemu-kvm)          1165.30            5.29         1170.59
10442 (qemu-kvm)          1853.42            5.30         1858.71
10614 (qemu-kvm)          1229.18            5.36         1234.54
11300 (qemu-kvm)          8102.38            5.33         8107.71
11609 (qemu-kvm)             0.02         1243.33         1243.35
11745 (qemu-kvm)             0.03         1632.27         1632.30
11950 (qemu-kvm)          1171.24            5.29         1176.53
13916 (qemu-kvm)             0.03         1197.76         1197.79
18752 (qemu-kvm)           106.37          174.97          281.34
19510 (qemu-kvm)             0.03         5140.26         5140.29
20728 (qemu-kvm)          1390.66            5.33         1395.98
24182 (qemu-kvm)          2172.11            5.33         2177.44
---------------  --------------- --------------- ---------------
Total                    17190.77         9425.81        26616.58

Per-node numastat info (in MBs):
                          Node 0          Node 1           Total
                 --------------- --------------- ---------------
Numa_Hit            3266639.70      1461553.41      4728193.11
Numa_Miss                 0.11            7.25            7.36
Numa_Foreign              7.25            0.11            7.36
Interleave_Hit           79.86           79.83          159.68
Local_Node          3263671.00      1461452.61      4725123.61
Other_Node             2968.81          108.05         3076.86
[root@rhel-h5 ~]#
```

Example – RHEL Hypervisor with numad

File Edit View Search Terminal Help

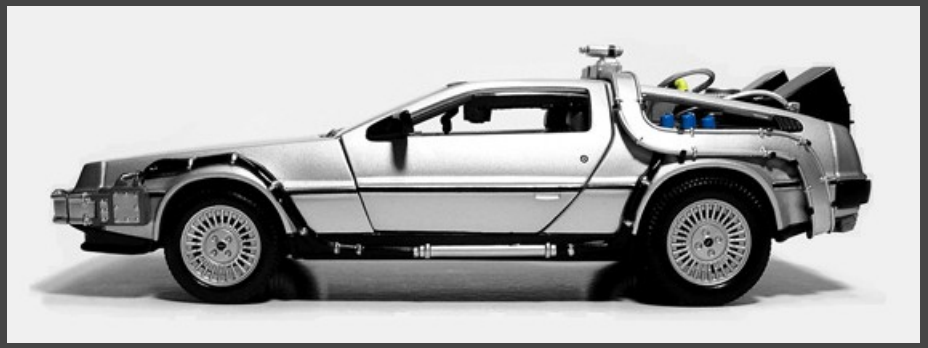
For the most part, VMs are no longer split between nodes.

```
[root@rhel-h5 ~]# numastat -n qemu

Per-node process memory usage (in MBs)
PID                         Node 0          Node 1           Total
---------------  --------------- --------------- ---------------
10308 (qemu-kvm)          1165.30            5.29         1170.59
10442 (qemu-kvm)          1853.42            5.30         1858.71
10614 (qemu-kvm)          1229.18            5.36         1234.54
11300 (qemu-kvm)          8102.38            5.33         8107.71
11609 (qemu-kvm)             0.02         1243.33         1243.35
11745 (qemu-kvm)             0.03         1632.27         1632.30
11950 (qemu-kvm)          1171.24            5.29         1176.53
13916 (qemu-kvm)             0.03         1197.76         1197.79
18752 (qemu-kvm)           106.37          174.97          281.34
19510 (qemu-kvm)             0.03         5140.26         5140.29
20728 (qemu-kvm)          1390.66            5.33         1395.98
24182 (qemu-kvm)          2172.11            5.33         2177.44
---------------  --------------- --------------- ---------------
Total                    17190.77         9425.81        26616.58

Per-node numastat info (in MBs):
                          Node 0          Node 1           Total
                 --------------- --------------- ---------------
Numa_Hit            3266639.70      1461553.41      4728193.11
Numa_Miss                 0.11            7.25            7.36
Numa_Foreign              7.25            0.11            7.36
Interleave_Hit           79.86           79.83          159.68
Local_Node            3263671.00      1461452.61      4725123.61
Other_Node               2968.81          108.05         3076.86
[root@rhel-h5 ~]#
```

More performance tuning?

Red Hat Enterprise Performance Tuning (RH442)
http://www.redhat.com/training/courses/rh442/

Building Your Own RPMs



Why do we package software?

Build Environment

- Setting up your initial build environment:
  - yum -y groupinstall “Development Tools”
  - yum -y install rpmdevtools

- As a **non-root** user run:

```
$ rpmdev-setuptree
$ ls ~/rpmbuild
BUILD  RPMS  SOURCES  SPECS  SRPMS
```

Build Environment

- Create a GPG key: gpg --gen-key
- gpg --export -a "Marty McFly" --armor > RPM-GPG-KEY-BTTF

- Add to ~/.rpmmacros
  - %_signature gpg
  - %_gpg_name ECAA997B (<-from gpg --list-keys)

.spec File - Preamble

```
Name:
Version:
Release:        1%{?dist}
Summary:

Group:
License:
URL:
Source0:
BuildRoot:      %(mktemp -ud %{_tmppath}/%{name}-%{version}-%
{release}-XXXXXX)

BuildRequires:
Requires:

%description
```

.spec File - Preamble

```
Name: screen
Version: 4.0.3
Release: 16%{?dist}
Summary: A screen manager that supports multiple logins on one terminal

Group: Applications/System
License: GPLv2+
URL: http://www.gnu.org/software/screen
BuildRoot: %{_tmppath}/%{name}-%{version}-%{release}-root-%(%
{__id_u} -n)
BuildRequires: ncurses-devel pam-devel libutempter-devel autoconf
texinfo
Requires(pre): /usr/sbin/groupadd
Requires(preun): /sbin/install-info
Requires(post): /sbin/install-info
Source0: ftp://ftp.uni-erlangen.de/pub/utilities/screen/screen-%
{version}.tar.gz
Source1: screen.pam
```

.spec File - Preamble

```
Patch1: screen-4.0.3-libs.patch
# Some tweaks of the default screenrc
Patch2: screen-4.0.3-screenrc.patch
Patch3: screen-4.0.3-configh.patch
Patch4: screen-4.0.3-stropts.patch
# Fixes potential buffer overflow when > 2^31 semicolons are passed.
Patch7: screen-4.0.1-args.patch
Patch11: screen-4.0.2-maxstr.patch
Patch12: screen-4.0.3-ipv6.patch
Patch13: screen-4.0.3-resize.patch
Patch14: screen-4.0.3-cc.patch

%description
The screen utility allows you to have multiple logins on just one
terminal. Screen is useful for users who telnet into a machine or are
connected via a dumb terminal, but want to use more than just one
login.
```

.spec File - %prep

- %setup -q

- From screen's spec:

```
%prep
%setup -q
%patch1 -p1 -b .libs
%patch2 -p1 -b .screenrc
%patch3 -p1 -b .configh
%patch4 -p1 -b .stropts
%patch7 -p0 -b .args
%patch11 -p1 -b .maxstr
%patch12 -p1 -b .ipv6
%patch13 -p2 -b .resize
%patch14 -p1 -b .cc
```

.spec File - %build

```
%configure
make %{?_smp_mflags}
```

- From screen's spec:

```
%build
autoconf

%configure \
        --enable-pam \
        --enable-colors256 \
        --enable-rxvt_osc \
        --enable-use-locale \
.........

make %{?_smp_mflags}
```

.spec - %files

%defattr(-,root,root,-)

%doc NEWS README doc/FAQ doc/README.DOTSCREEN COPYING

%attr(2755,root,screen) %{_bindir}/screen

%{_mandir}/man1/screen.*

%{_infodir}/screen.info*

%{_datadir}/screen

%attr(775,root,screen) %{_localstatedir}/run/screen

%config(noreplace) %{_sysconfdir}/screenrc

%config(noreplace) %{_sysconfdir}/pam.d/screen

.spec - %files

Not sure which files are included?

Run `rpmbuild -bb [specfile]` to generate a list.

.spec File - Scripts

- %pre
- %post – run ldconfig
- %preun & postun – typically “undo” anything from the pre/post
- DO NOT USE SCRIPTS TO CHEAT!!
  - tar -zxvf software.tgz && configure && make && make install
  - checkout from CVS, SVN, git <--really bad idea
- DO NOT REQUIRE END USER INPUT!!!

.spec File - %changelog

The format matters!

* Tue Jan 25 2011 Emmett Brown <doc@bttf.com> - 4.0.3-16
- fix potential problems for Common Criteria certification (#665103)
- Repair space-time continuum (#88mpr)

* Fri Sep 25 2009 Biff Tannen <biff@bttr.com> - 4.0.3-15
- acquire generic sports almanac from future (#515055)
- achieve great wealth via gambling (#523647, #506256, #492729)
- avoid manure container (#1985)
- suppress install-info errors (#515999)

Compiling - rpmbuild

- Use `rpmbuild` to test each section of the spec file

Build options with [<specfile> | <tarball> | <source package>]:

-bp	build through %prep (unpack sources and apply patches) from <specfile>
-bc	build through %build (%prep, then compile) from <specfile>
-bi	build through %install (%prep, %build, then install) from <specfile>
-bl	verify %files section from <specfile>
-ba	build source and binary packages from <specfile>
-bb	build binary package only from <specfile>
-bs	build source package only from <specfile>

Test, Sign, & Deploy

- rpm --resign [your rpm]
- rpm --checksig [rpm]
- On your systems import your GPG Key
  - rpm --import RPM-GPG-KEY-BTTF

Resources

- Reference existing SRPMs
- Thomas Cameron's “RED HAT NETWORK POWER USER TIPS AND TRICKS PACKAGING SOFTWARE”
  - http://www.youtube.com/watch?v=K_A6U2nWntE
- Fedora RPM guide:
  - http://docs.fedoraproject.org/en-US/Fedora_Draft_Documentation/0.1/html/RPM_Guide/
- Fedora Packaging Guidlines:
  - http://fedoraproject.org/wiki/Packaging:Guidelines
- Labs: http://people.redhat.com/~smcbrien
  - Download the handout and software

More RPM Building?

Red Hat System Administration III (**RH255**)
http://www.redhat.com/training/courses/rh255/

Red Hat Enterprise Deployment and Systems Management (**RH401**)
http://www.redhat.com/training/courses/rh401/

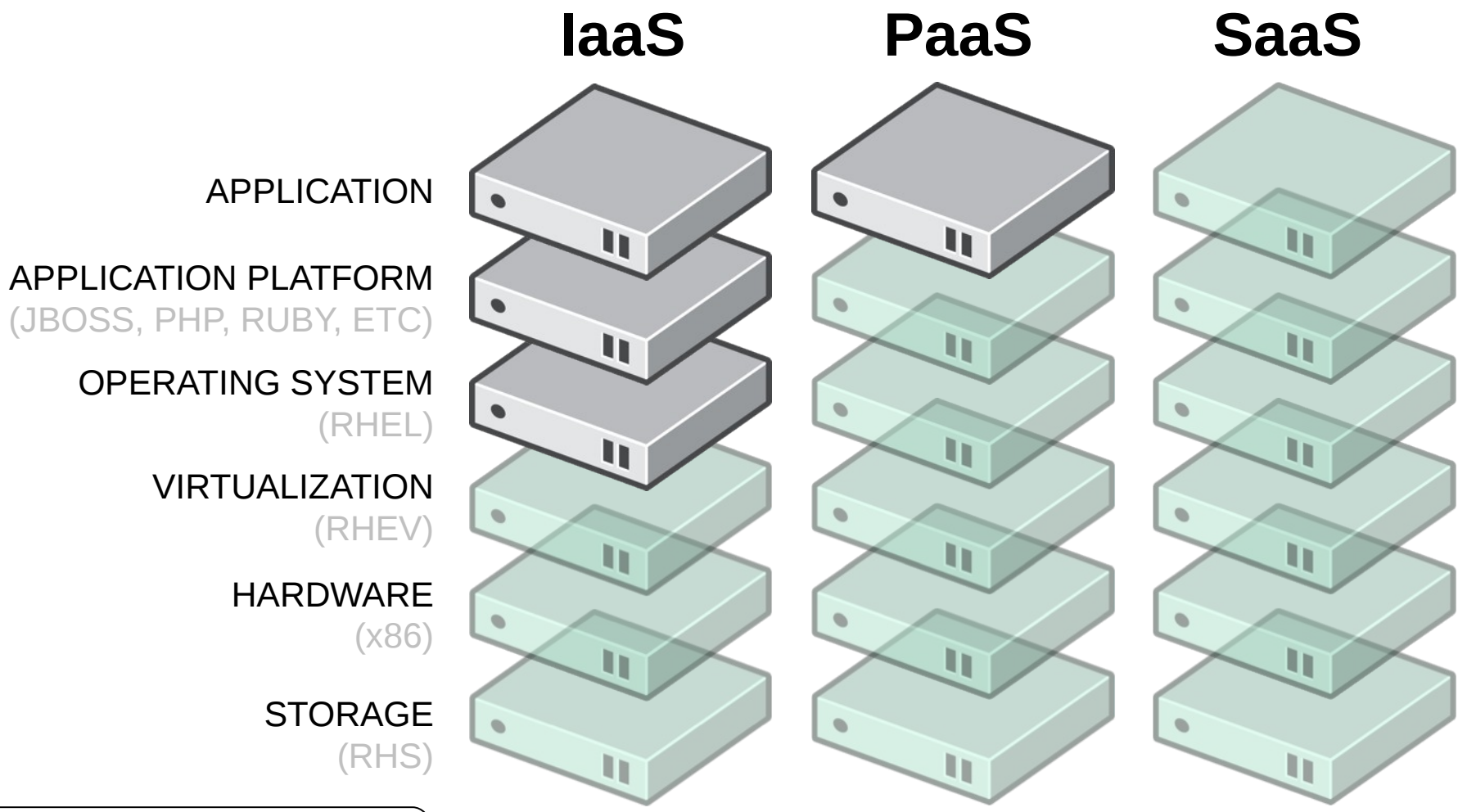
SELINUX FOR MERE MORTALS

OPENSHIFT

redhat.

PAAS = PLATFORM AS A SERVICE

CLOUD COMPUTING AND PAAS



CRAFTWORK → MASS PRODUCTION

Physical

How to Build an App:

1. Have Idea
2. Get Budget
3. Submit hardware acquisition request
4. Wait
5. Get Hardware
6. Rack and Stack Hardware
7. Install Operating System
8. Install Operating System Patches/Fix-Packs
9. Create user Accounts
10. Deploy framework/appserver
11. Deploy testing tools
12. Code
13. Test
14. Configure Prod servers (and buy them if needed)
15. Push to Prod

• Launch

1. Order more servers to meet demand
2. Wait…
3. Deploy new servers
4. Etc.

Virtualized

How to Build an App:

1. Have Idea
2. Get Budget
3. Submit VM Request request
4. Wait
5. Deploy framework/appserver
6. Deploy testing tools
7. Code
8. Test
9. Configure Prod VMs
10. Push to Prod
11. Launch
12. Request More Prod VMs to meet demand
13. Wait
14. Deploy app to new VMs
15. Etc.

With PaaS

How to Build an App:

1. **Have Idea**
2. **Get Budget**
3. **Code**
4. **Test**
5. **Launch**
6. **Automatically Scale**

*"The use of Platform-as-a-Service technologies will enable **IT organizations** to become more agile and **more responsive to the business needs**." –Gartner**

OPENSHIFT STRATEGY

Public Cloud Service

OPENSHIFT ONLINE
by Red Hat

OPENSHIFT STRATEGY

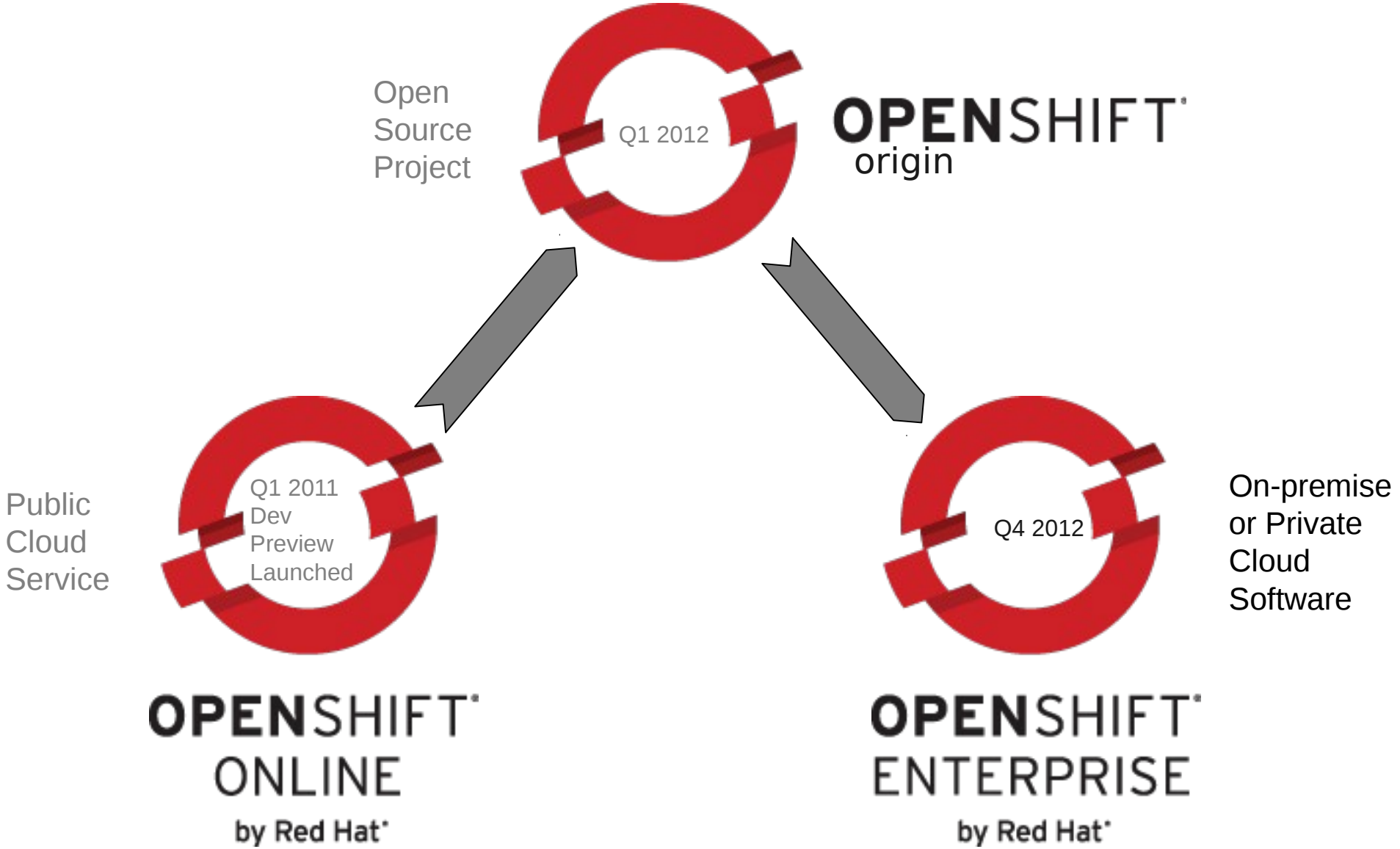
Open Source Project

OPENSHIFT origin

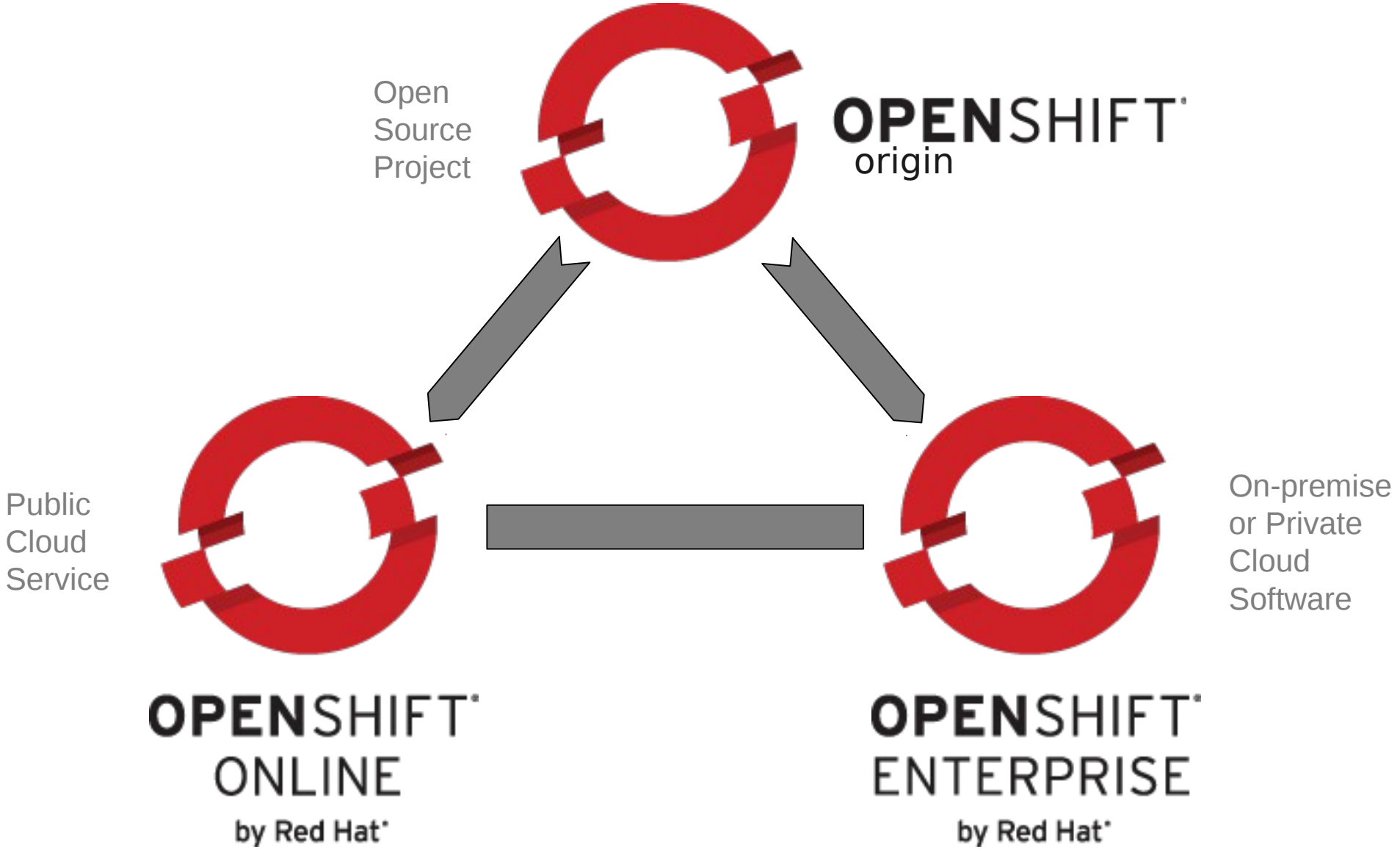
Public Cloud Service

OPENSHIFT ONLINE by Red Hat

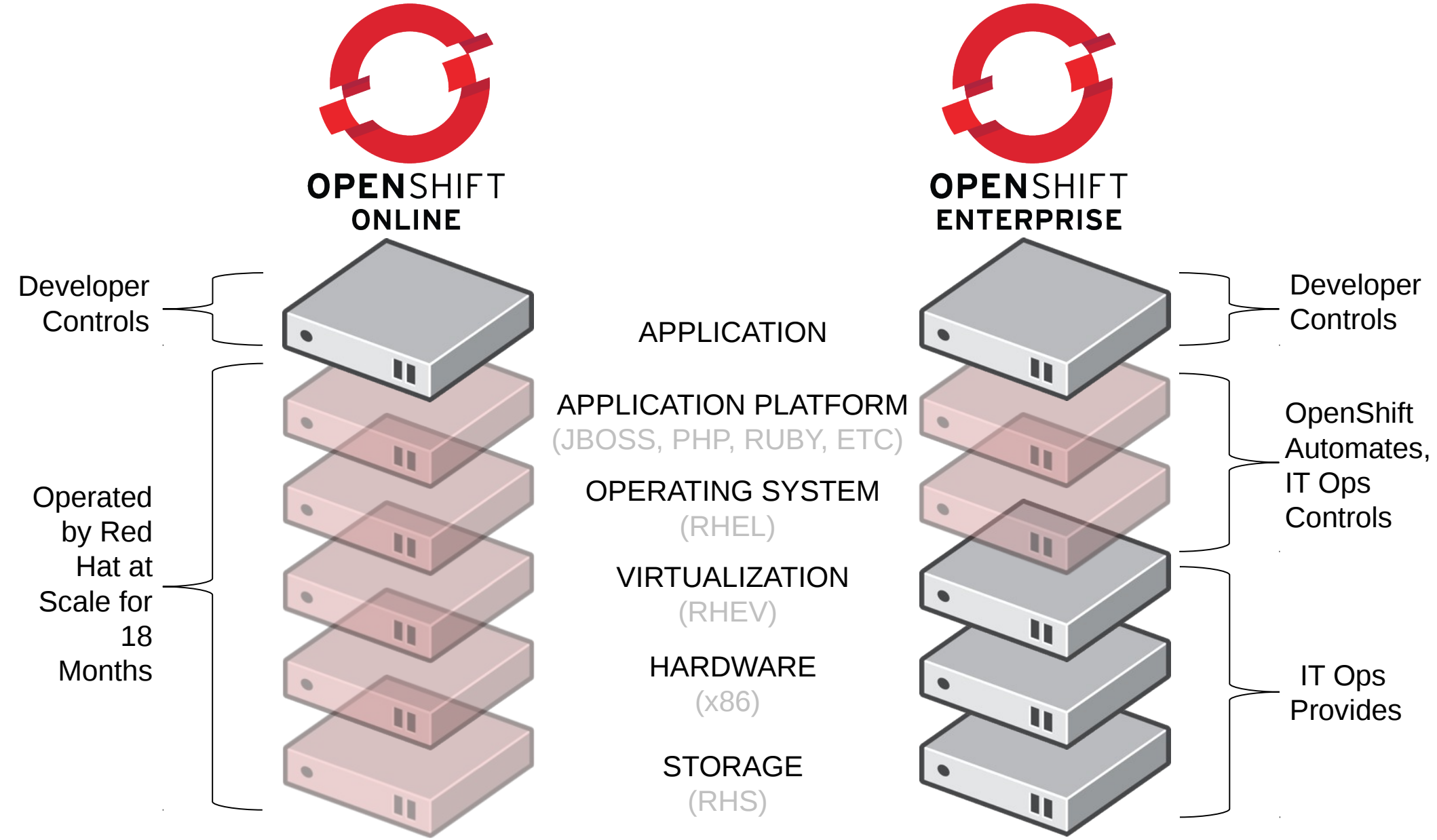
OPENSHIFT STRATEGY



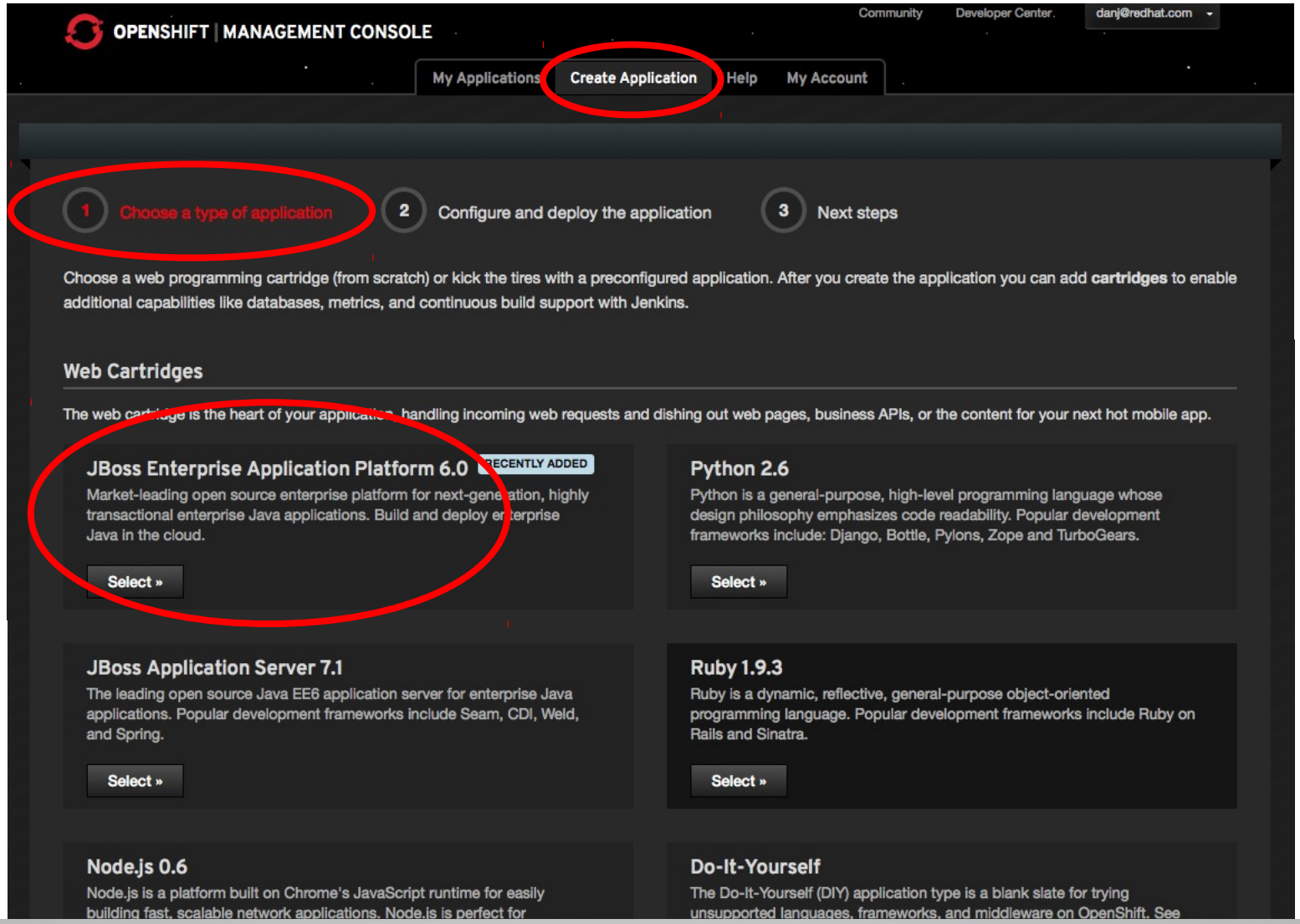
OPENSHIFT STRATEGY



HOW TO USE OPENSHIFT



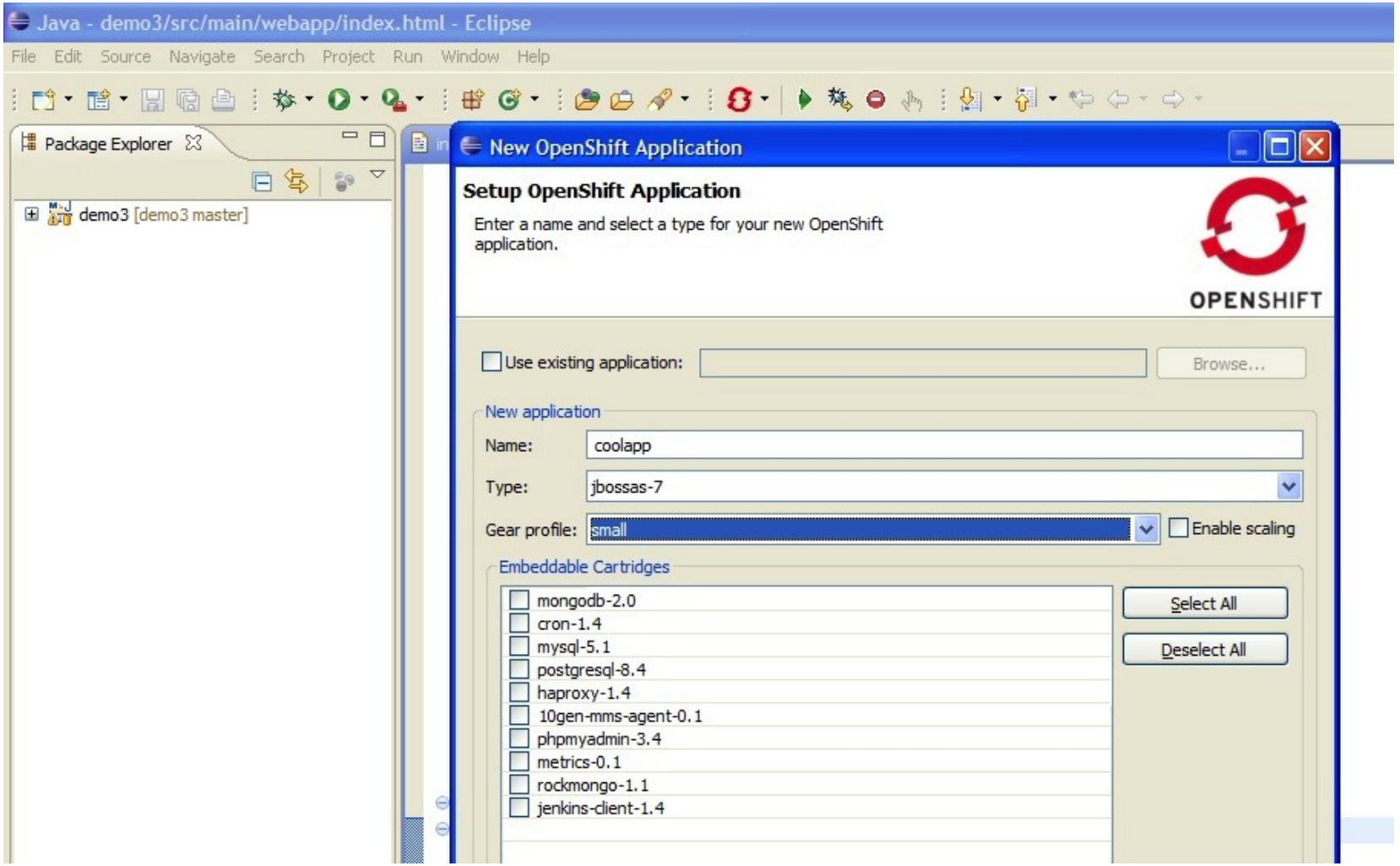
RESPONSIVE WEB CONSOLE



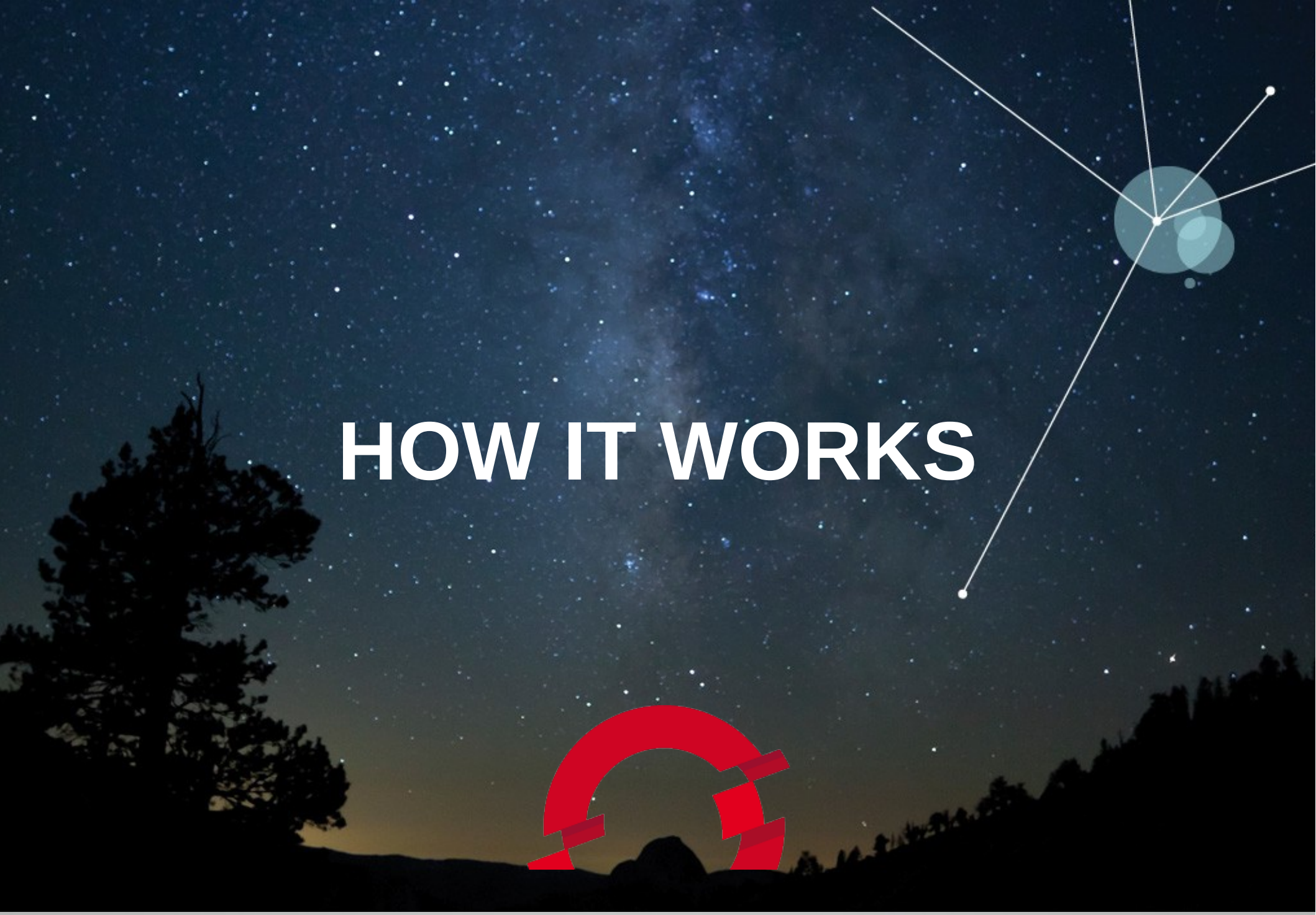
CLI? OF COURSE.

1. Create App
   ```
   rhc app create -a javasample -t jbossas-7
   ```
2. Add MongoDB
   ```
   rhc app cartridge add -a javasample –c mongodb-2.0
   ```
3. Add an EAR file to your deployments directory
   ```
   cd javasample

   cp /path/to/ear/earfilename.ear ./deployments
   ```
1. Add the EAR file to git
   ```
   git add ./deployments/earfilename.ear
   ```
2. Push your code
   ```
   git push
   ```
3. Done

ECLIPSE, TOO.


Java - demo3/src/main/webapp/index.html - Eclipse
File Edit Source Navigate Search Project Run Window Help
Package Explorer
demo3 [demo3 master]
New OpenShift Application
Setup OpenShift Application
Enter a name and select a type for your new OpenShift application.
OPENSHIFT
Use existing application:
Browse...
New application
Name:
coolapp
Type:
jbossas-7
Gear profile:
small
Enable scaling
Embeddable Cartridges
mongodb-2.0
cron-1.4
mysql-5.1
postgresql-8.4
haproxy-1.4
10gen-mms-agent-0.1
phpmyadmin-3.4
metrics-0.1
rockmongo-1.1
jenkins-client-1.4
Select All
Deselect All

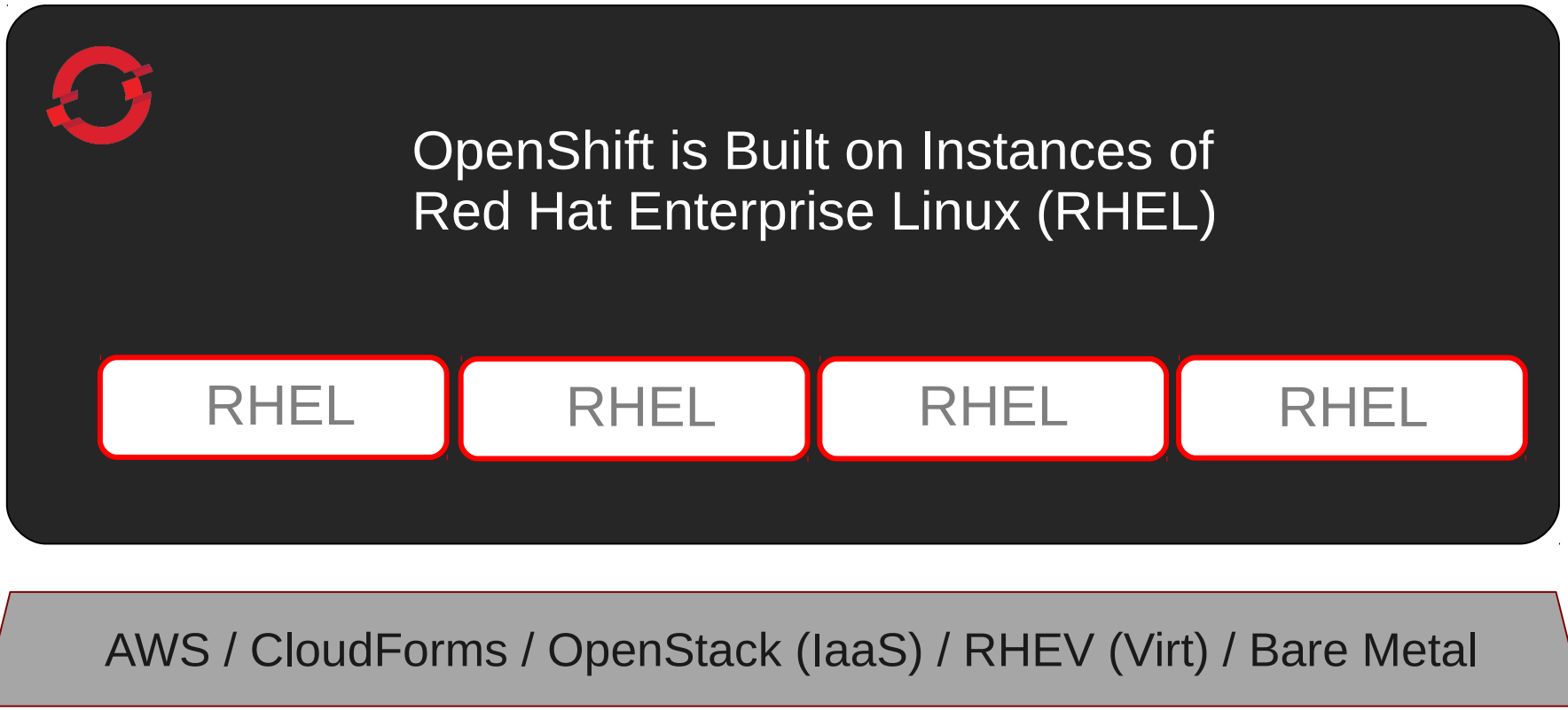
EVERYTHING YOU ALREADY USE.


HOW IT WORKS

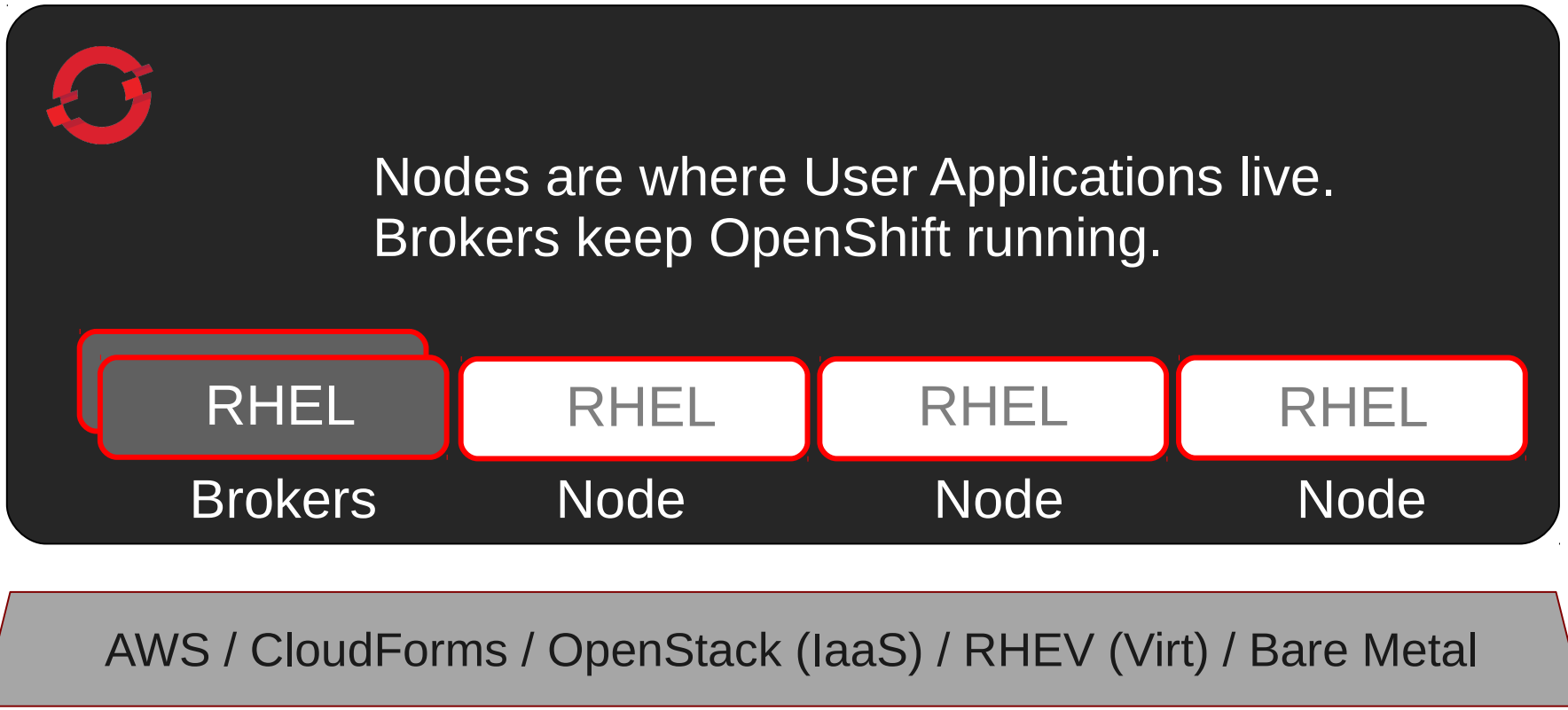
YES, WE STILL HAVE INFRASTRUCTURE.

AWS / CloudForms / OpenStack (IaaS) / RHEV (Virt) / Bare Metal

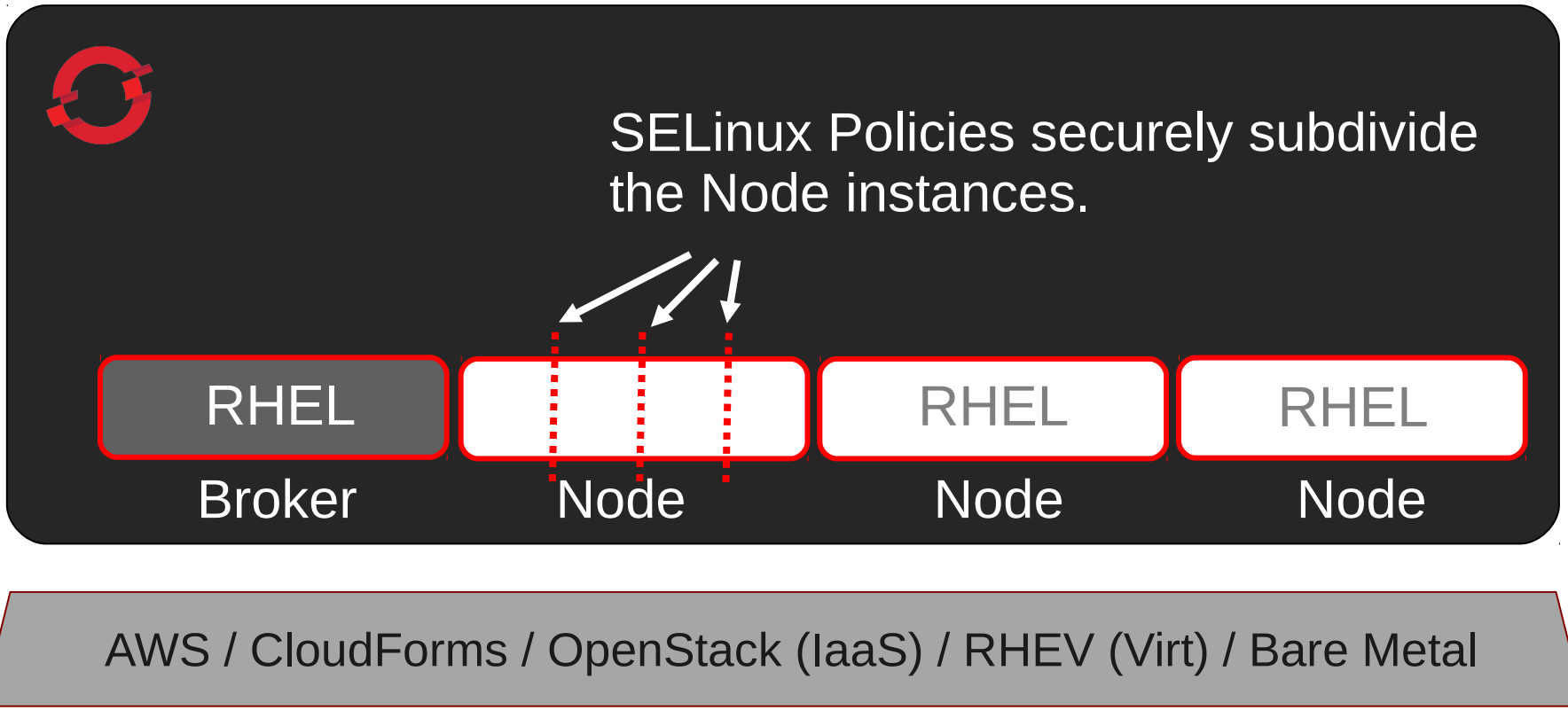
RHEL IS THE FOUNDATION.



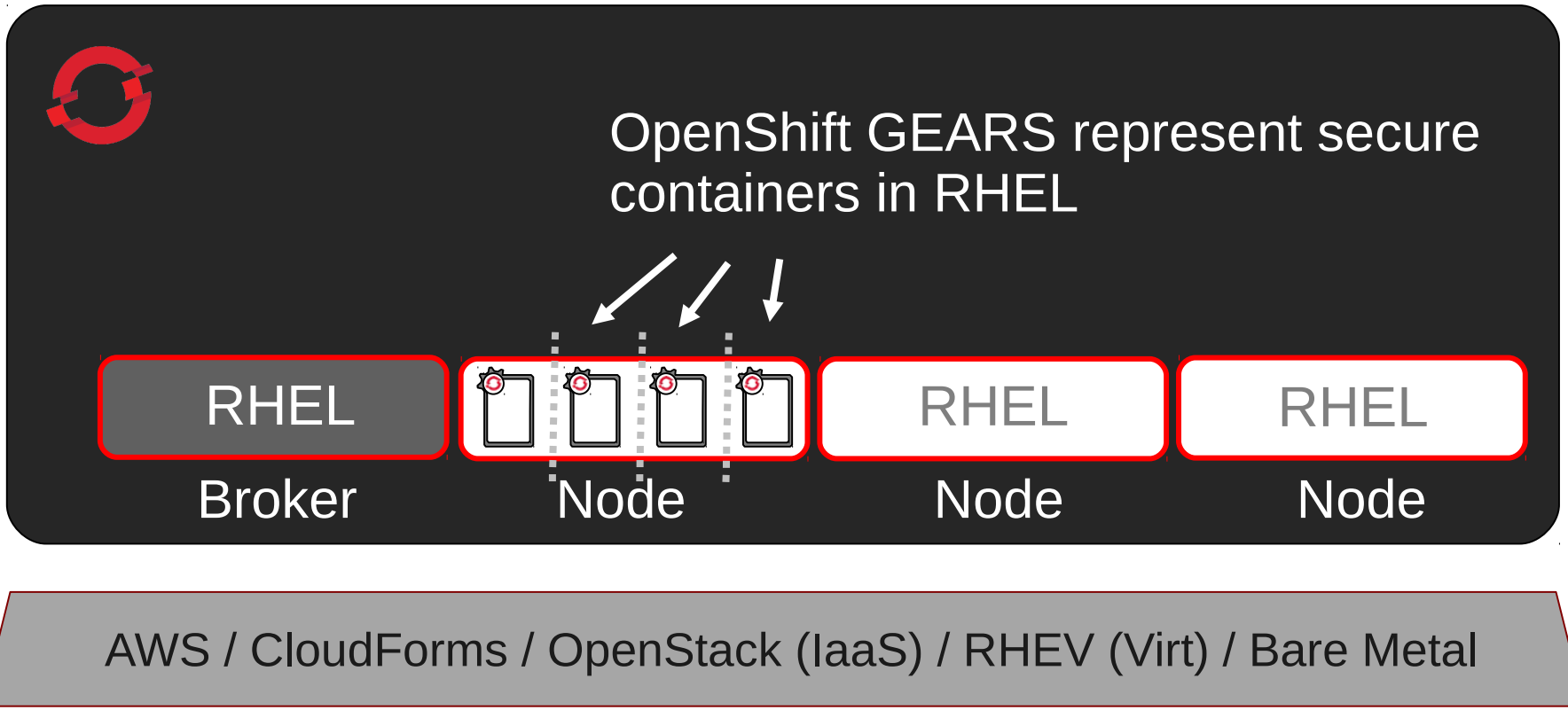
A BROKER MANAGES NODES.



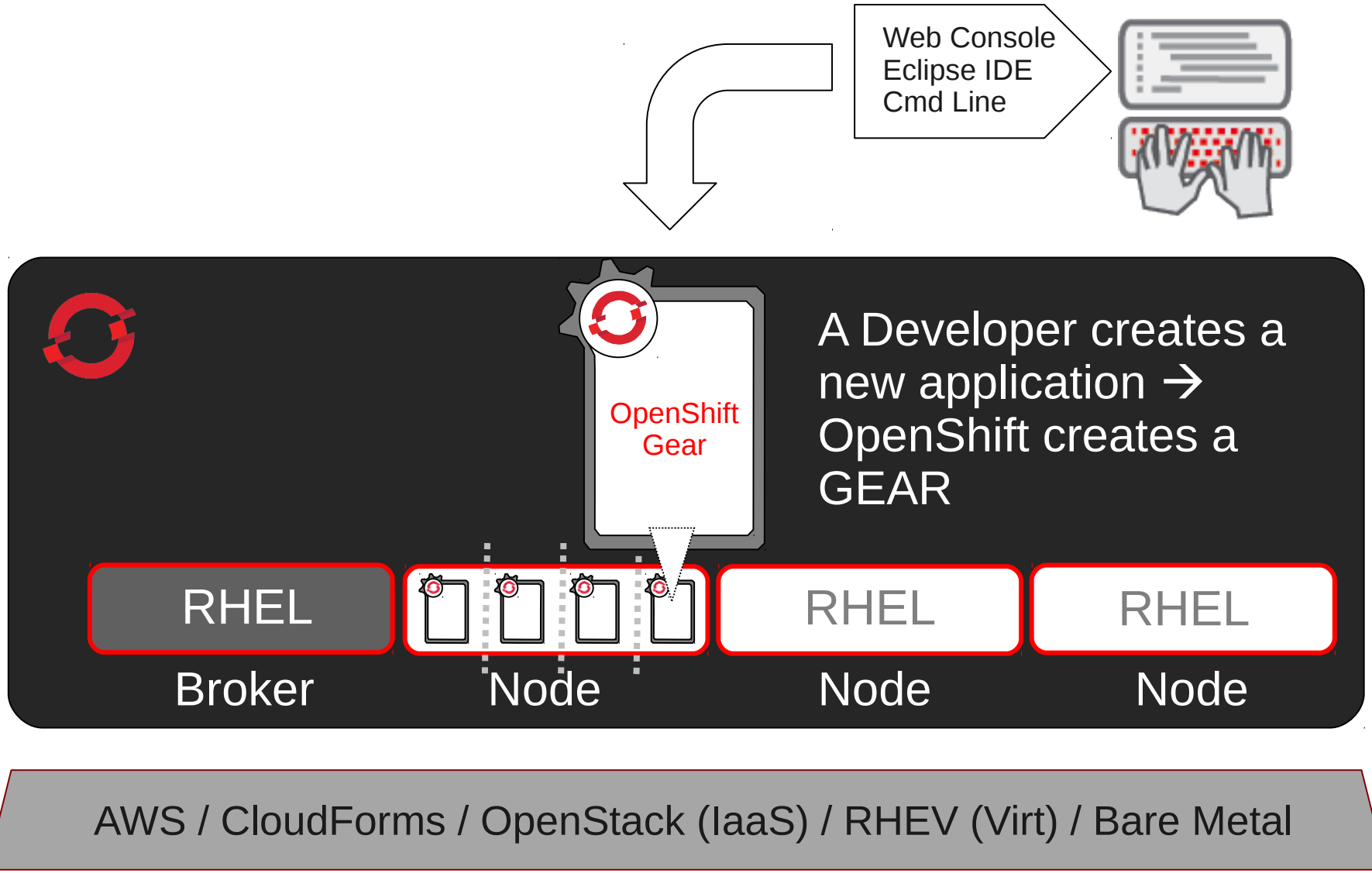
RHEL GIVES US MULTI-TENANCY.



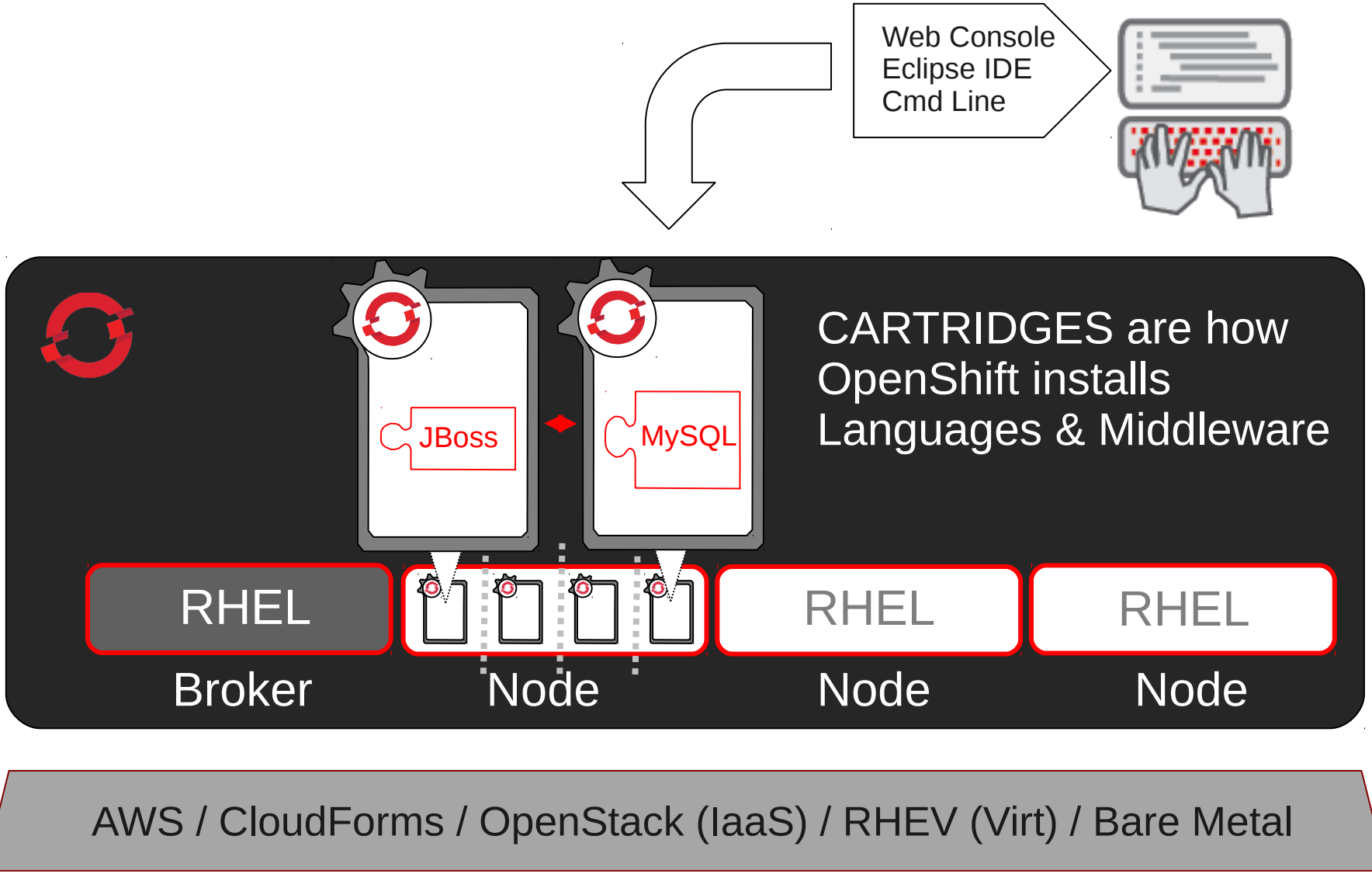
GEARS, NOT SERVERS.



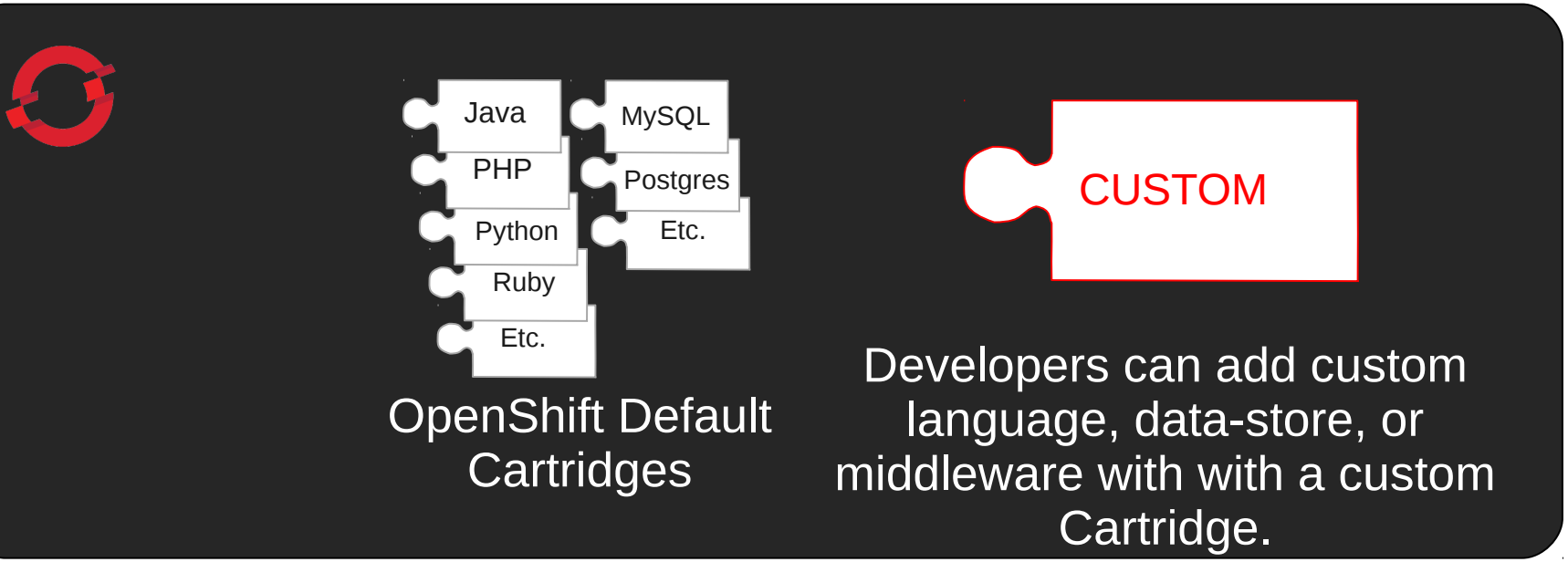
WORKFLOW


Web Console
Eclipse IDE
Cmd Line
OpenShift Gear
A Developer creates a new application → OpenShift creates a GEAR
RHEL
RHEL
RHEL
Broker
Node
Node
Node
AWS / CloudForms / OpenStack (IaaS) / RHEV (Virt) / Bare Metal

CARTRIDGES LIVE IN GEARS

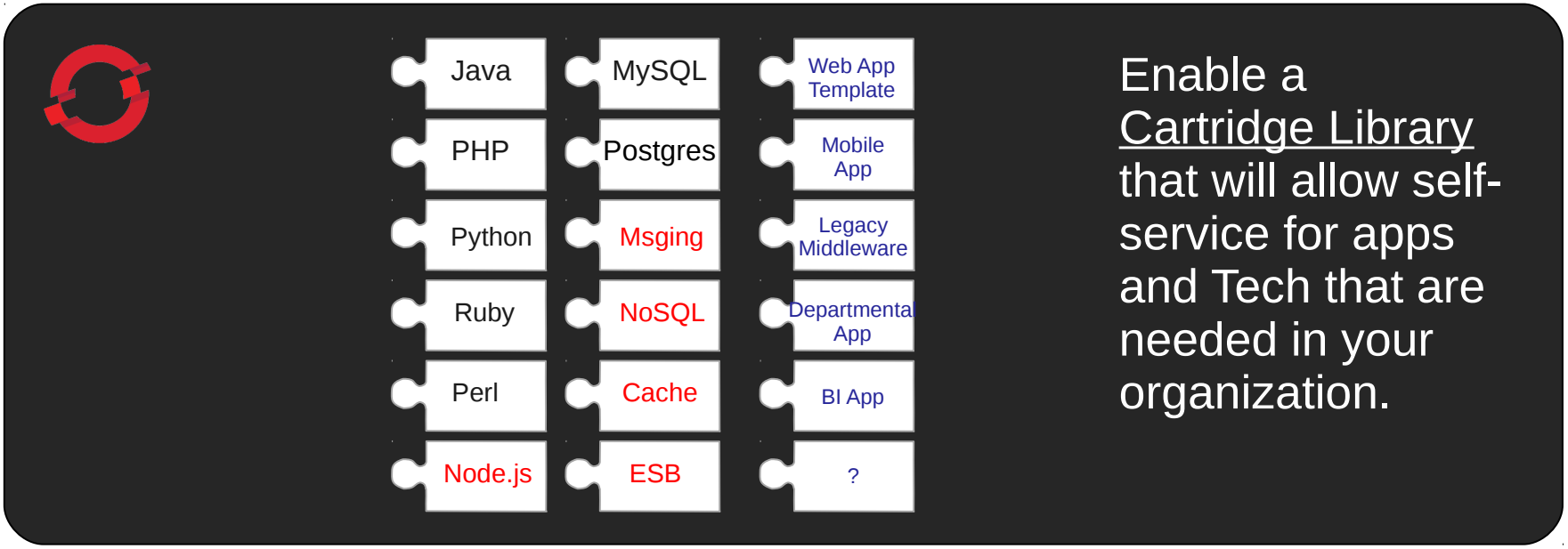


YES, YOU CAN BUILD YOUR OWN.



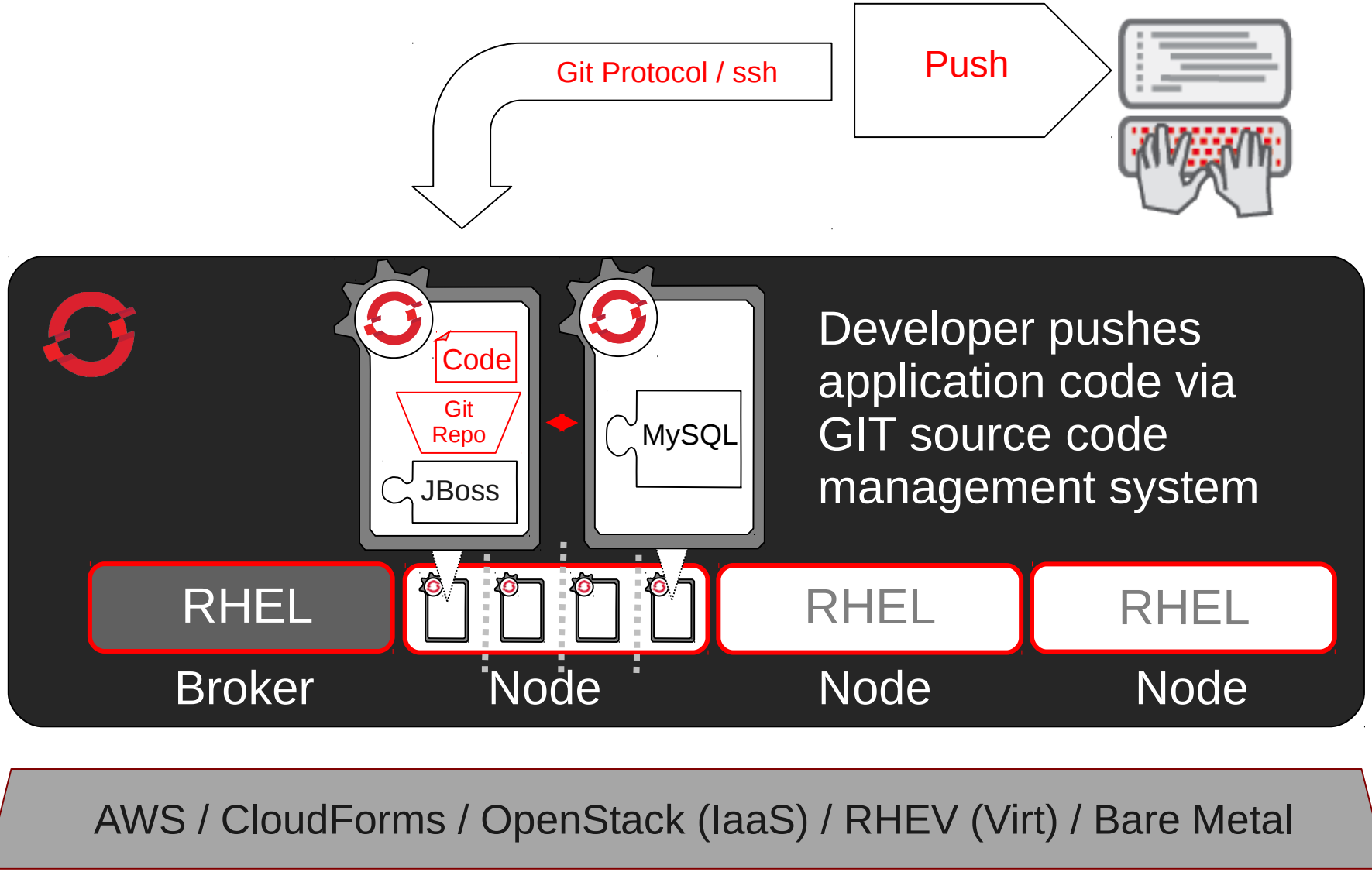
AWS / CloudForms / OpenStack (IaaS) / RHEV (Virt) / Bare Metal

QUICKSTARTS

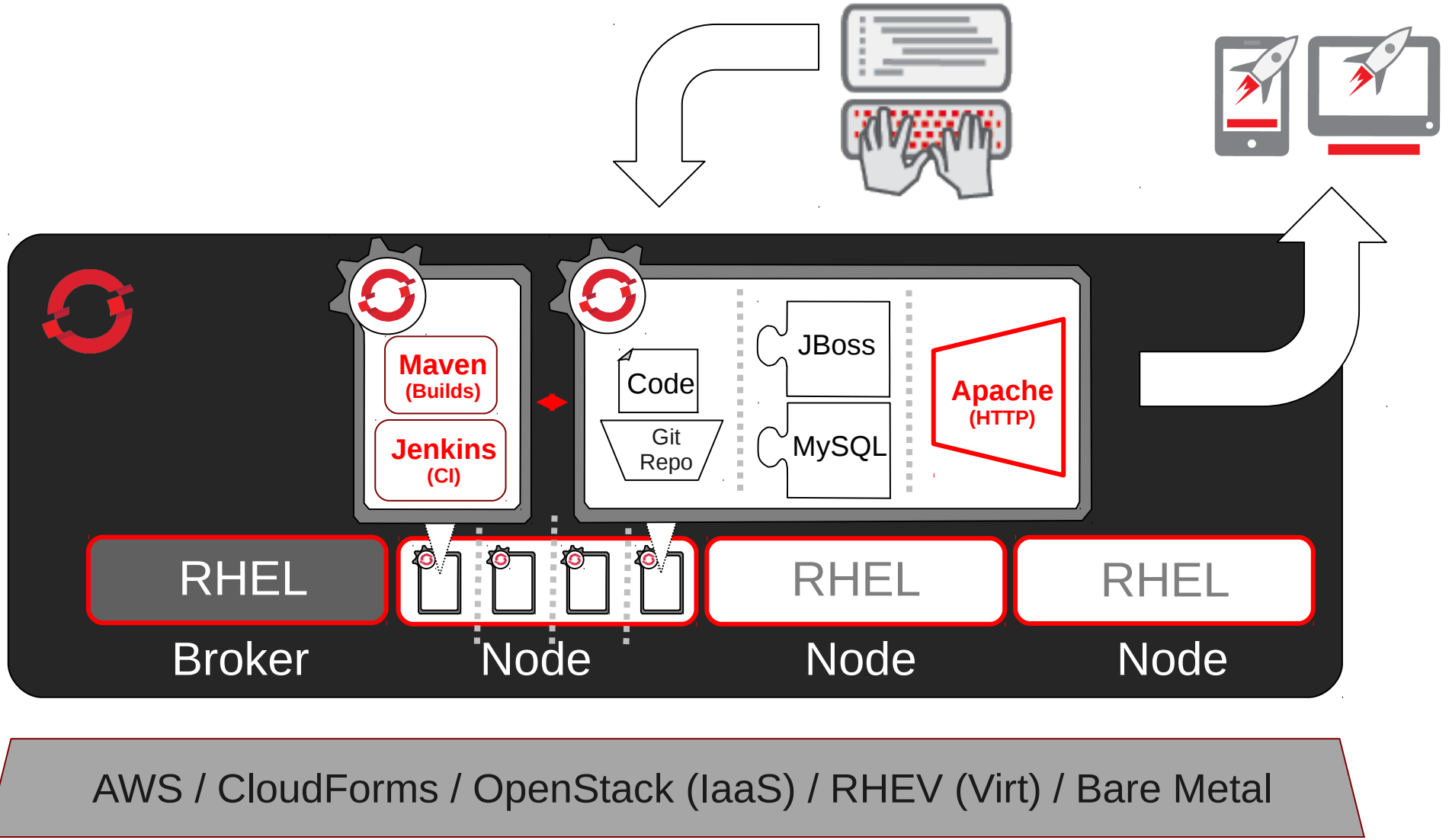


AWS / CloudForms / OpenStack (IaaS) / RHEV (Virt) / Bare Metal

GIT? YUP.

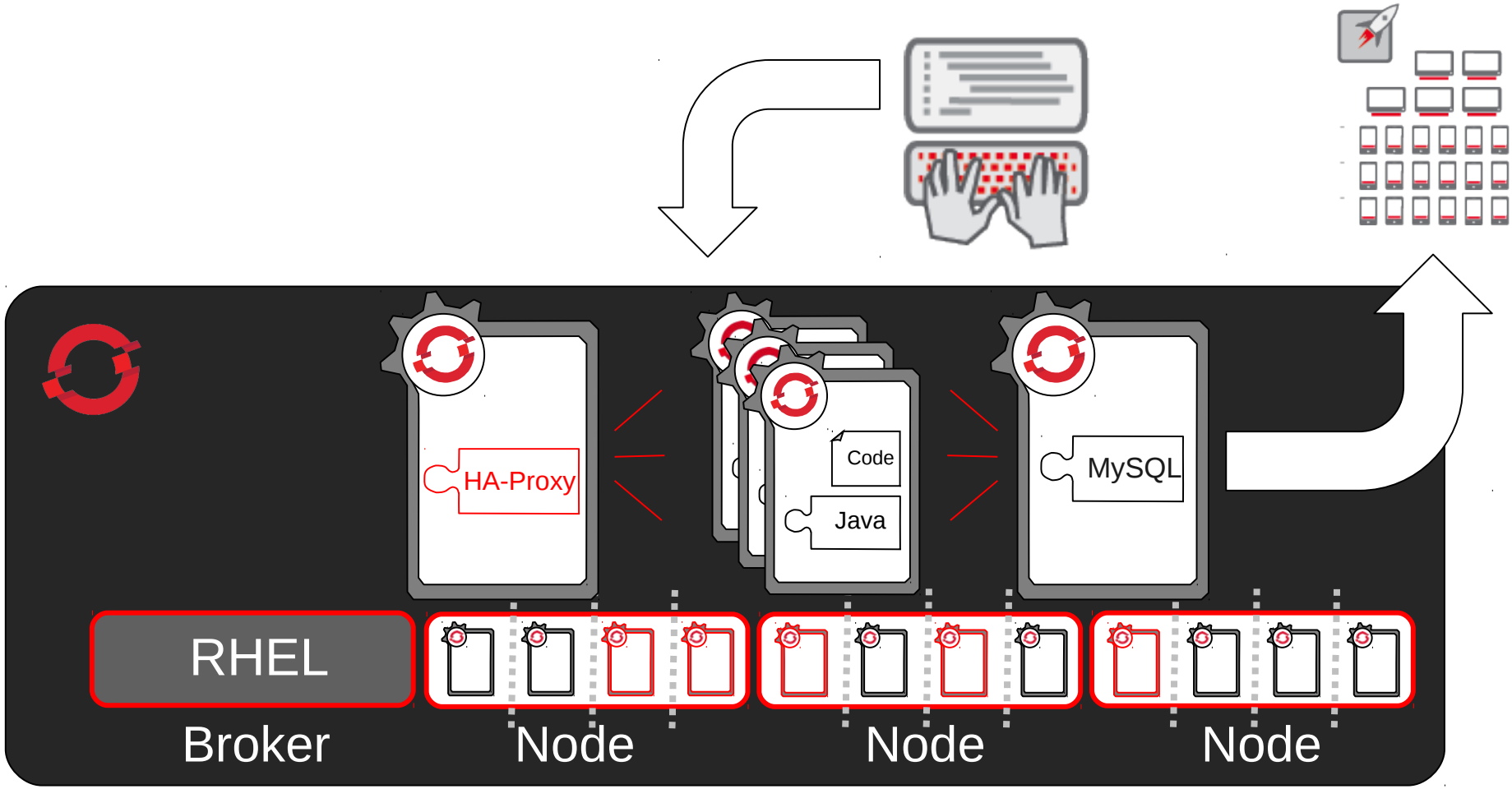

Git Protocol / ssh
Push
Code
Git Repo
JBoss
MySQL
Developer pushes application code via GIT source code management system
RHEL
RHEL
RHEL
Broker
Node
Node
Node
AWS / CloudForms / OpenStack (IaaS) / RHEV (Virt) / Bare Metal

AUTOMATE BUILD, TEST, PUSH.



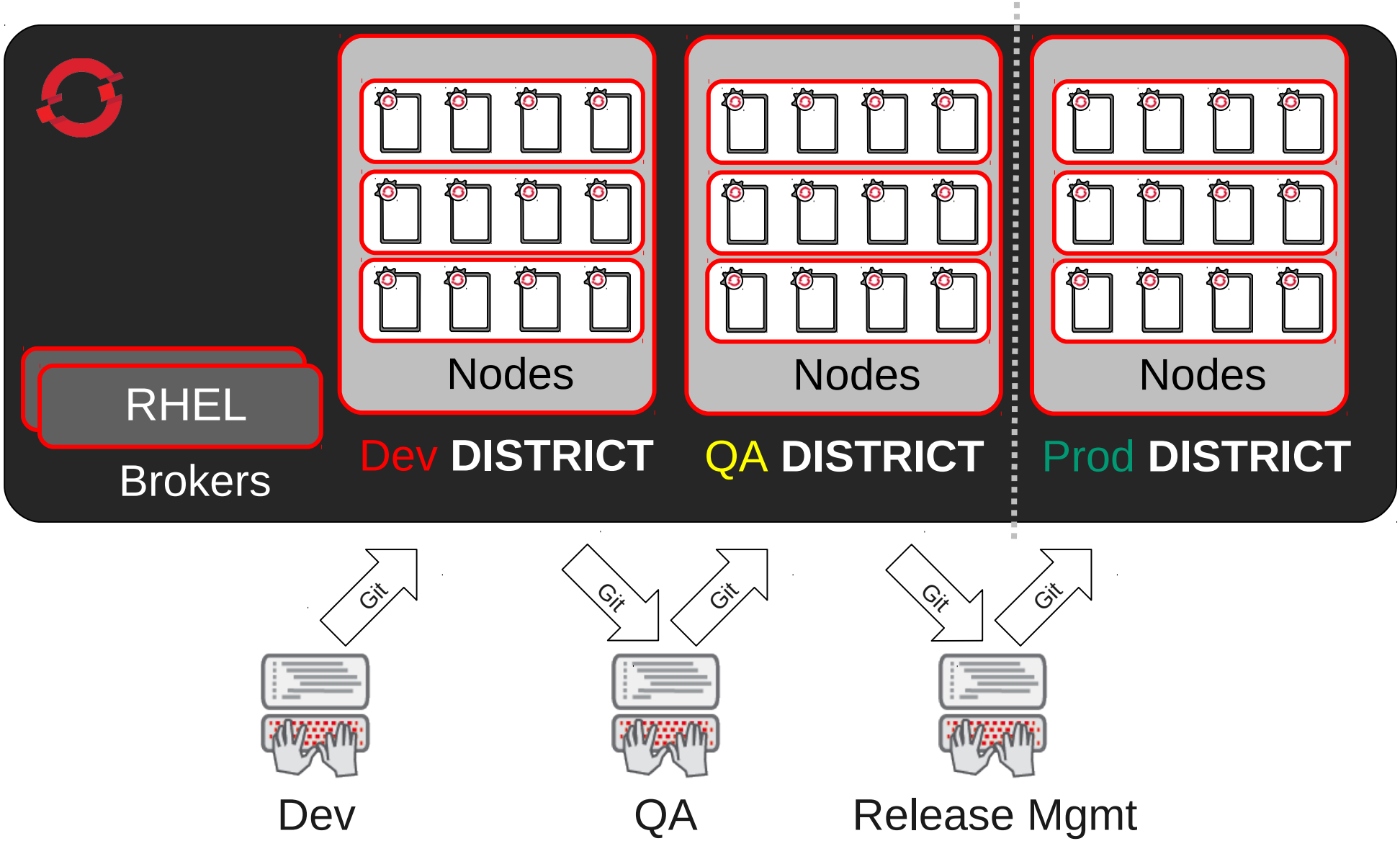
AWS / CloudForms / OpenStack (IaaS) / RHEV (Virt) / Bare Metal

AUTOMATED SCALING

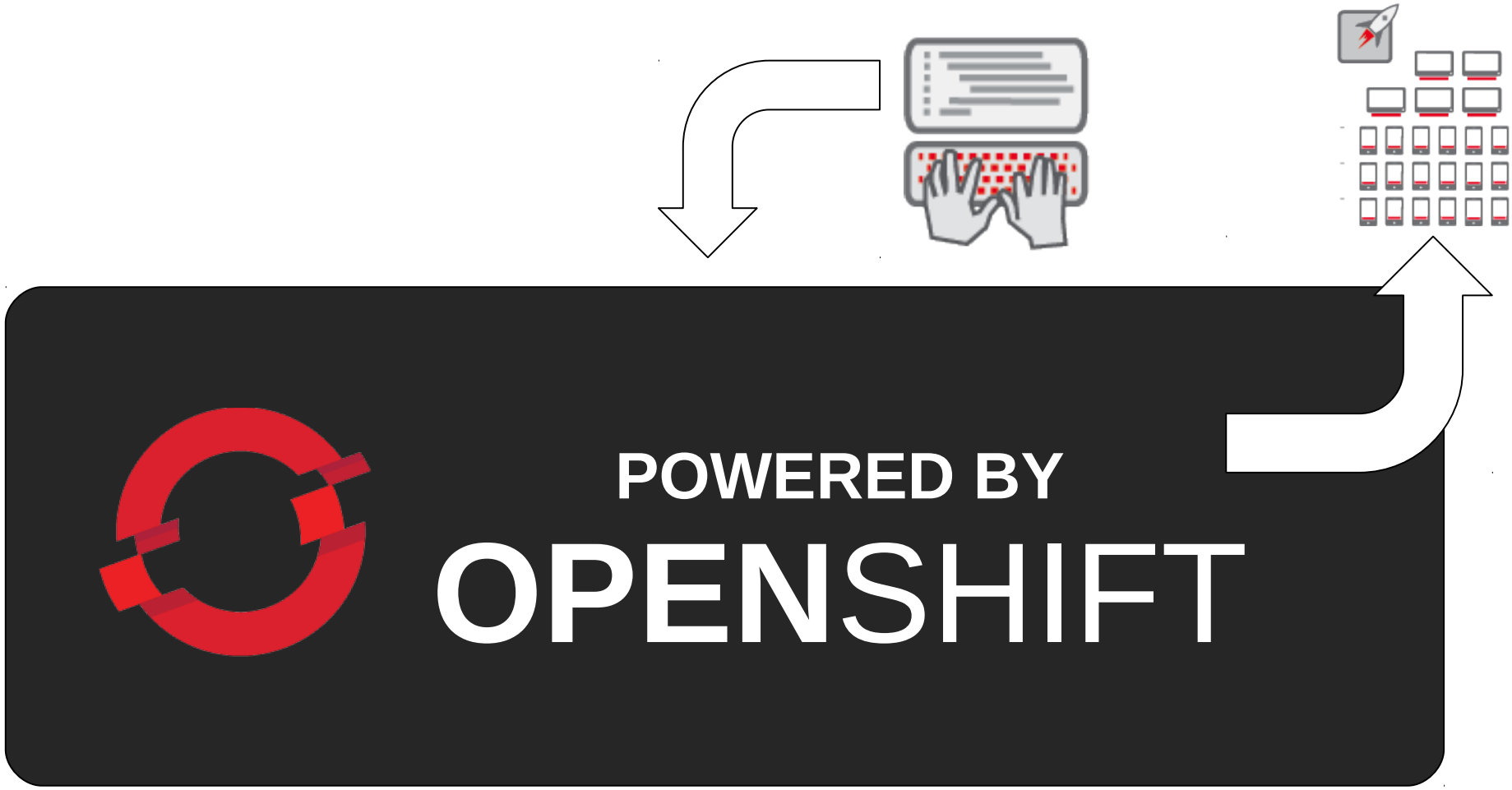


AWS / CloudForms / OpenStack (IaaS) / RHEV (Virt) / Bare Metal

Real-world App Dev
– Multi Environments, Single PaaS

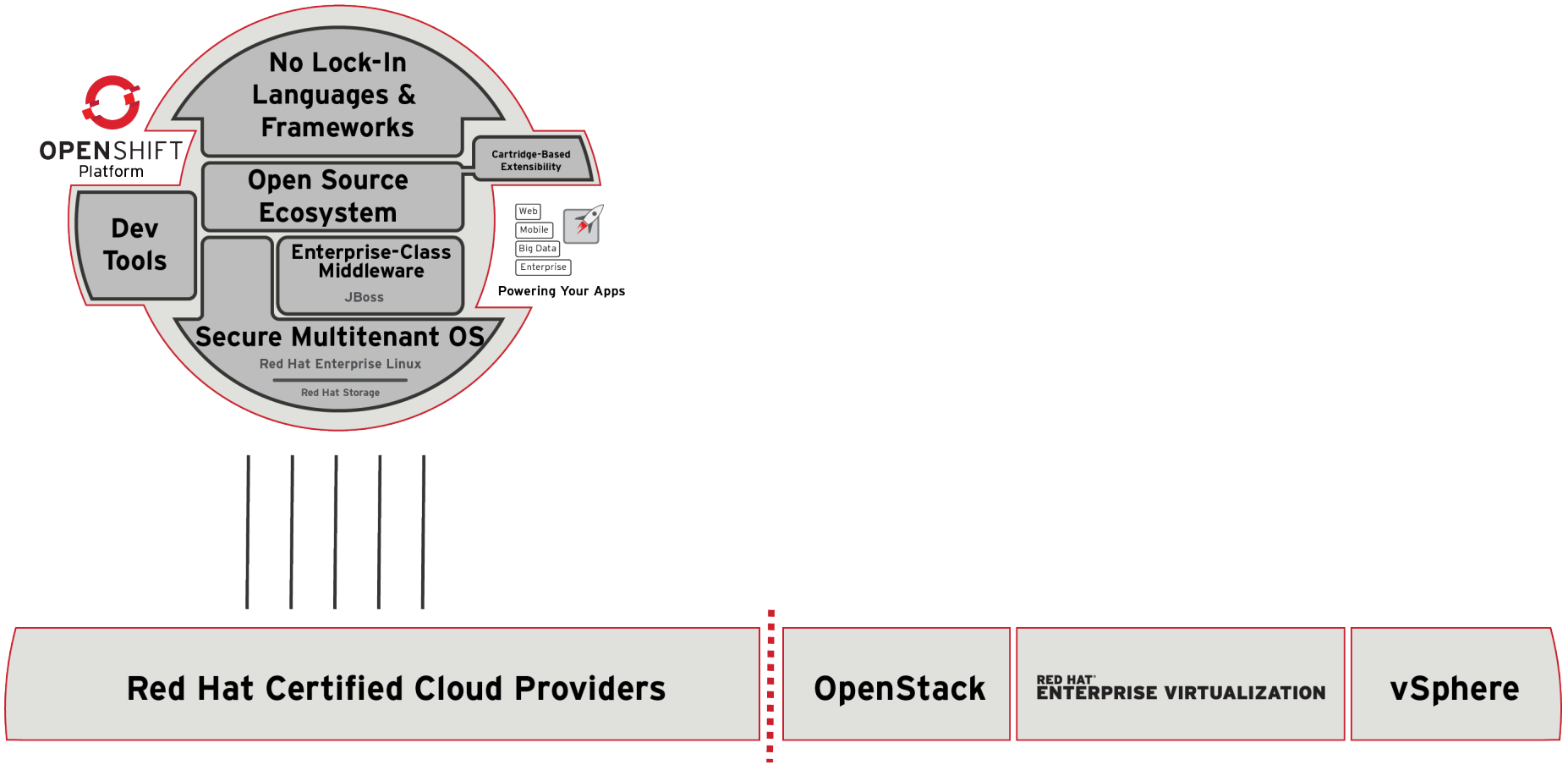


IT MANUFACTURING

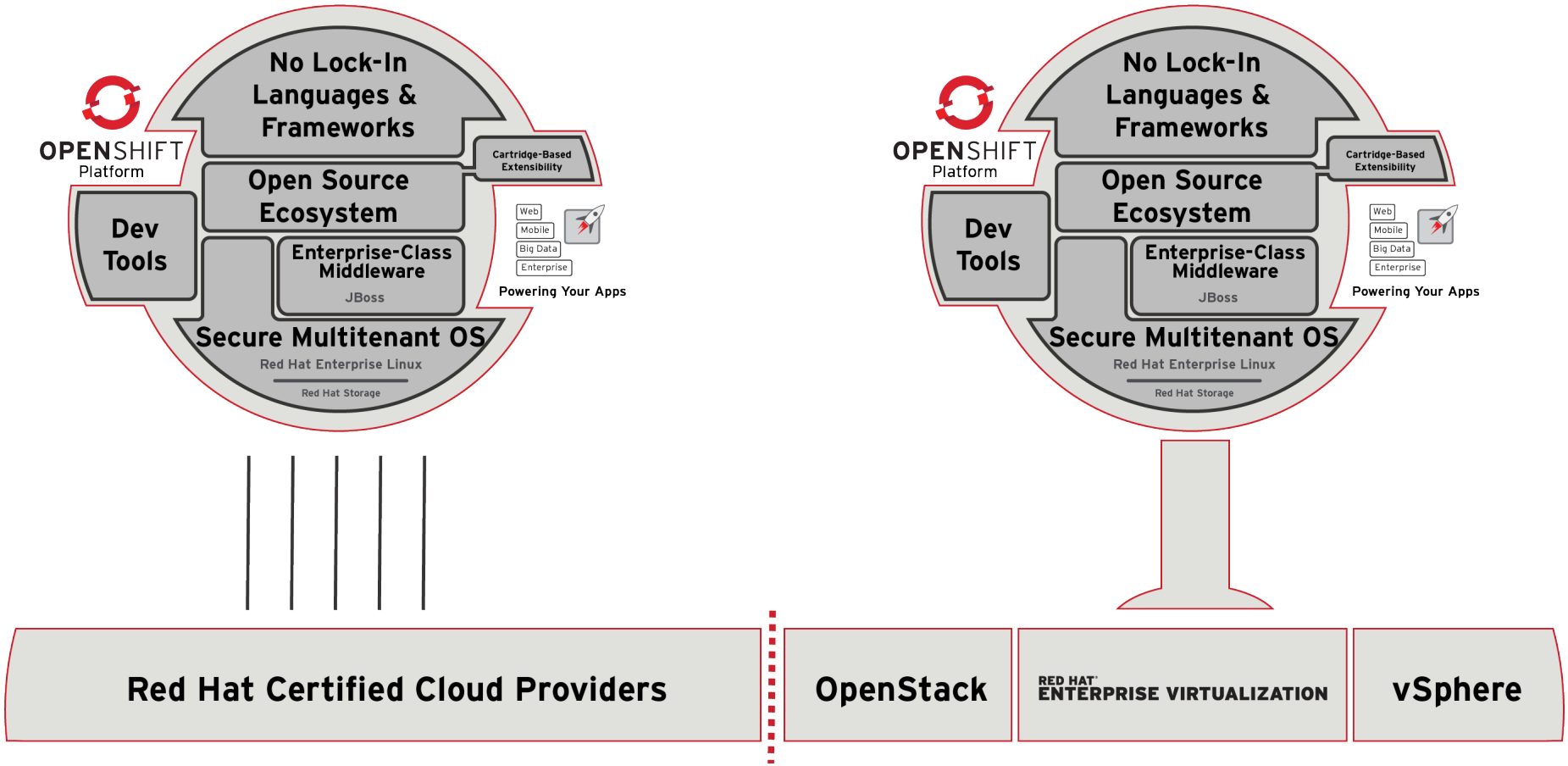


AWS / CloudForms / OpenStack (IaaS) / RHEV (Virt) / Bare Metal

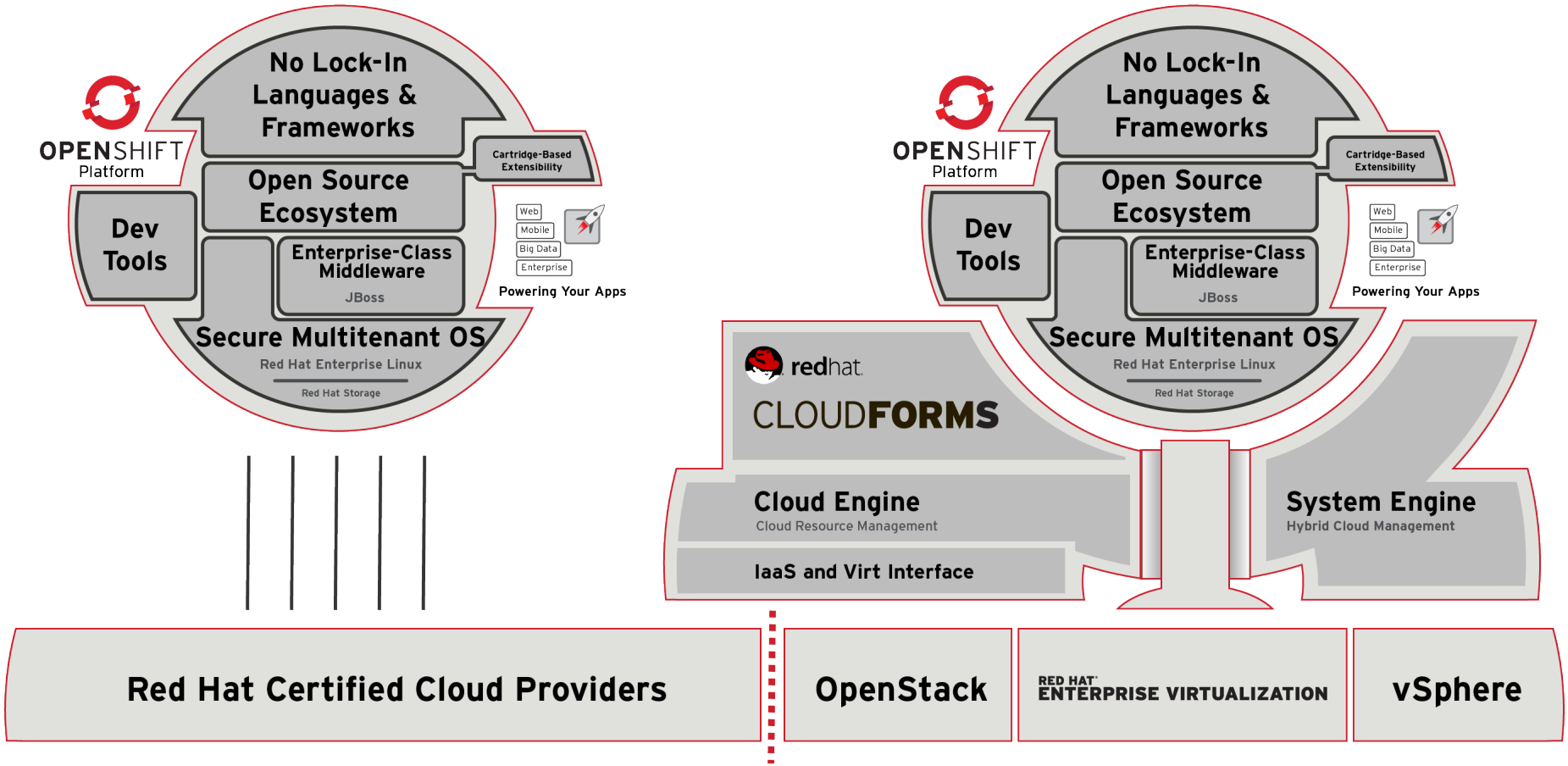
WHEREVER YOU WANT.



WHEREVER YOU WANT.

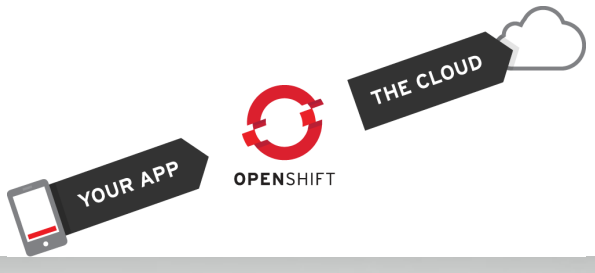


WHEREVER YOU WANT.



SO WHAT?

1. **Strength.** OpenShift is built on Red Hat work you know and love.
2. **Freedom.** In OpenShift, work the way you want.
   - Choice of Interface: Web Console, Command-line, or IDE
   - Choice of Middleware: Java(EE6), Ruby, Node.js, PHP, Python, etc.
   - Choice of Cloud: Public, Private, or Hybrid Cloud
   - Choice of Elasticity: Automatic application scaling when needed
1. **Openness.** OpenShift's open source software stack ensures application portability and No Lock-In.



OPENSHIFT ORIGIN

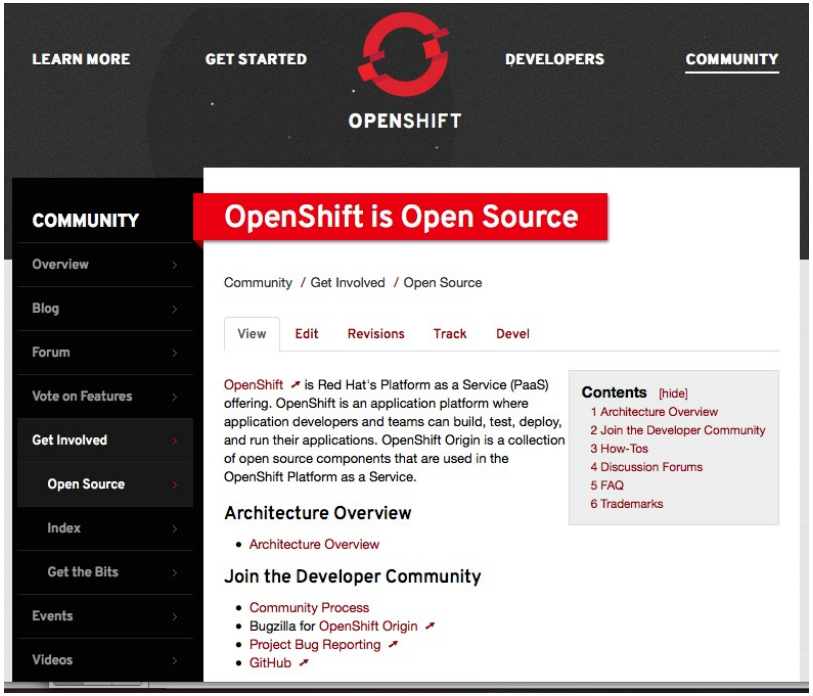
https://openshift.redhat.com/community/open-source
https://github.com/openshift
https://github.com/openshift/origin-server/cartridges

The upstream project for the OpenShift PaaS platform

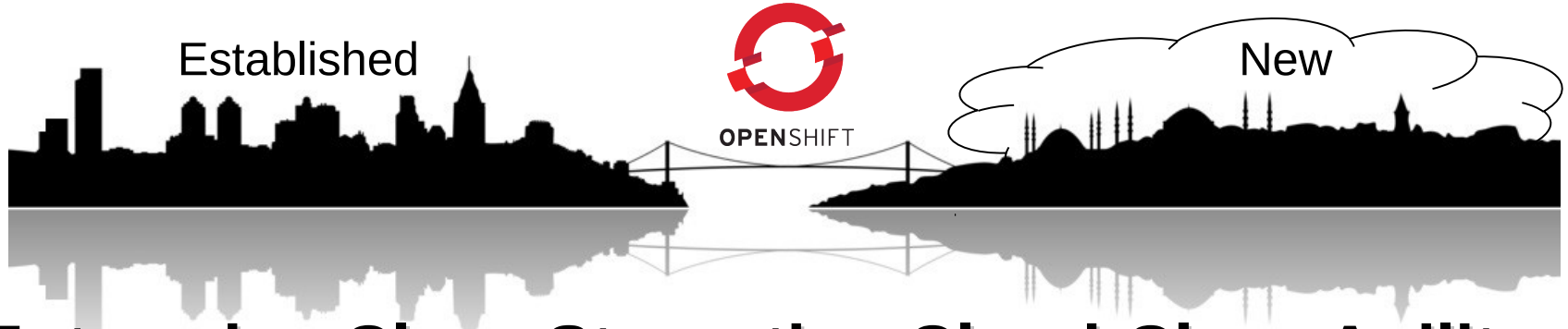
Apache 2.0 License

Available as:
Source, RPMs
.ISO, LiveCD (run your own)

IRC, email, forums



OPENSHIFT IS A BRIDGE.



Enterprise-Class Strength

- Enterprise Java EE6 via JBoss
- Multi-tenancy and Security via Red Hat Enterprise Linux
- Jenkins, Maven, Git
- Auto-Scaling
- On-Premise, Hosted, or Hybrid

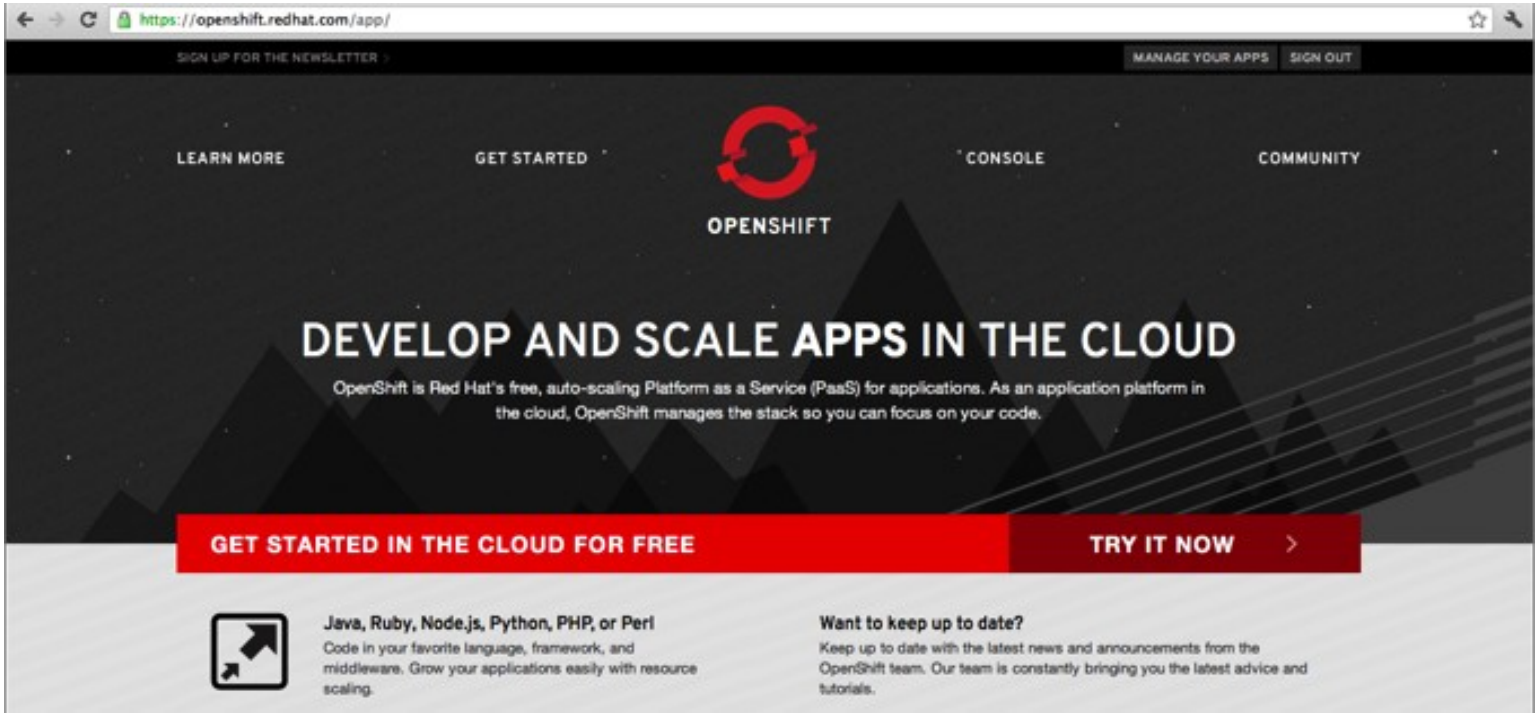
Cloud-Class Agility

- Designed for No Lock-In
- Polyglot with Java, Ruby, PHP, Perl, Python
- Mobile and Responsive Web
- REST and Javascript

OpenShift = Open Hybrid PaaS

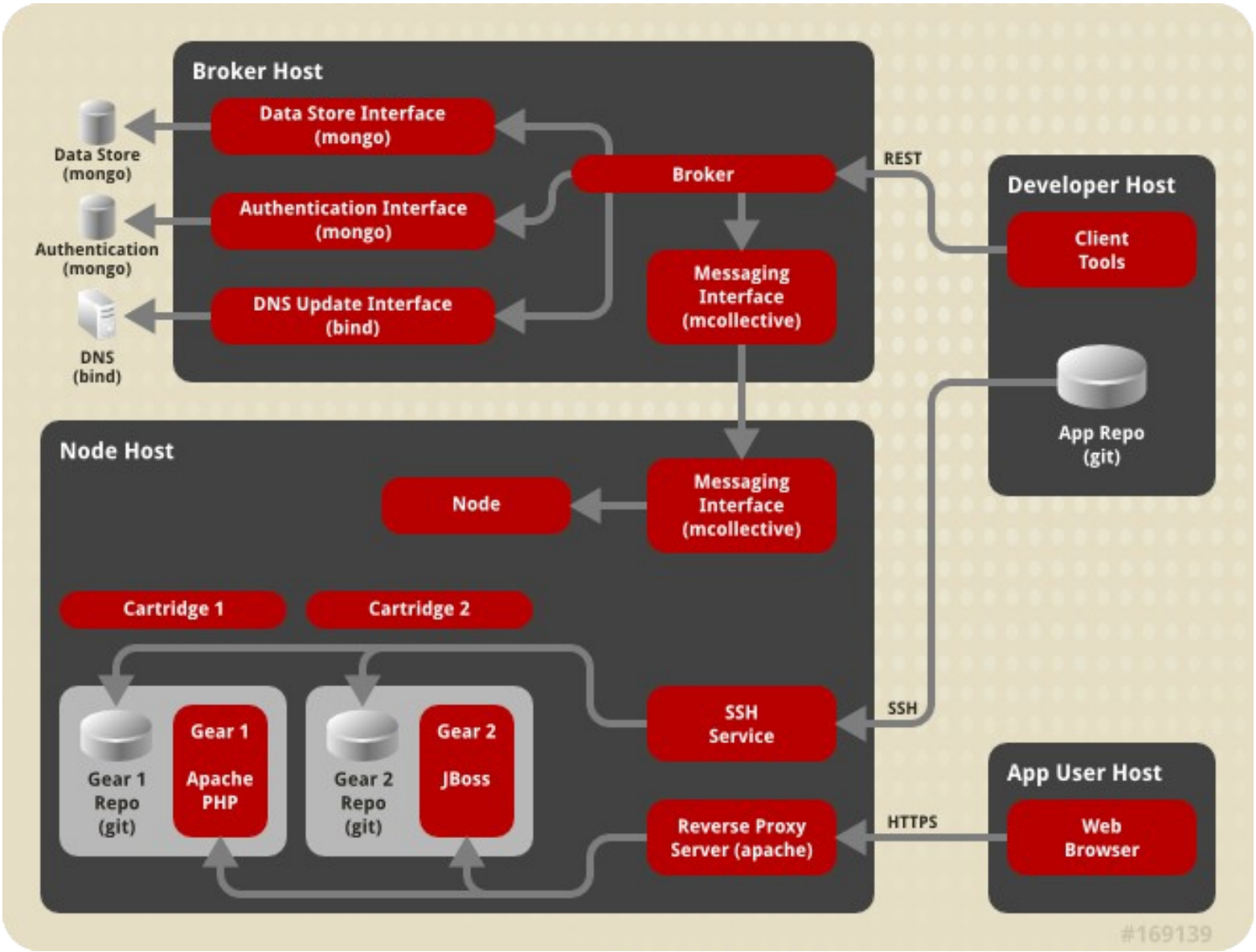
GETTING STARTED

- Deploy Apps to the **OpenShift OnLine** Developer Preview
- Request an Evaluation of **OpenShift Enterprise**
- Join the **OpenShift Origin** Open Source Project community

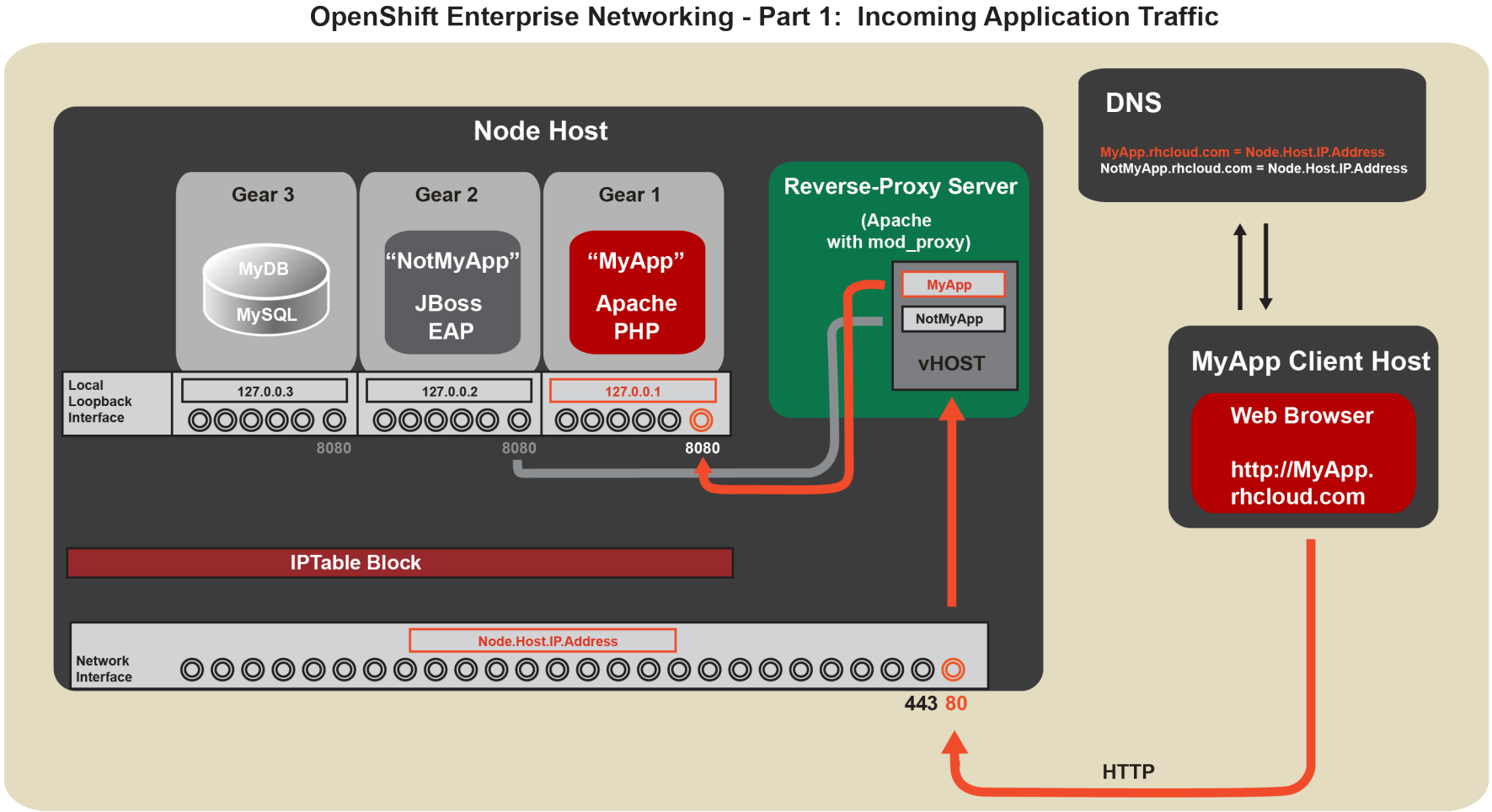


http://openshift.redhat.com

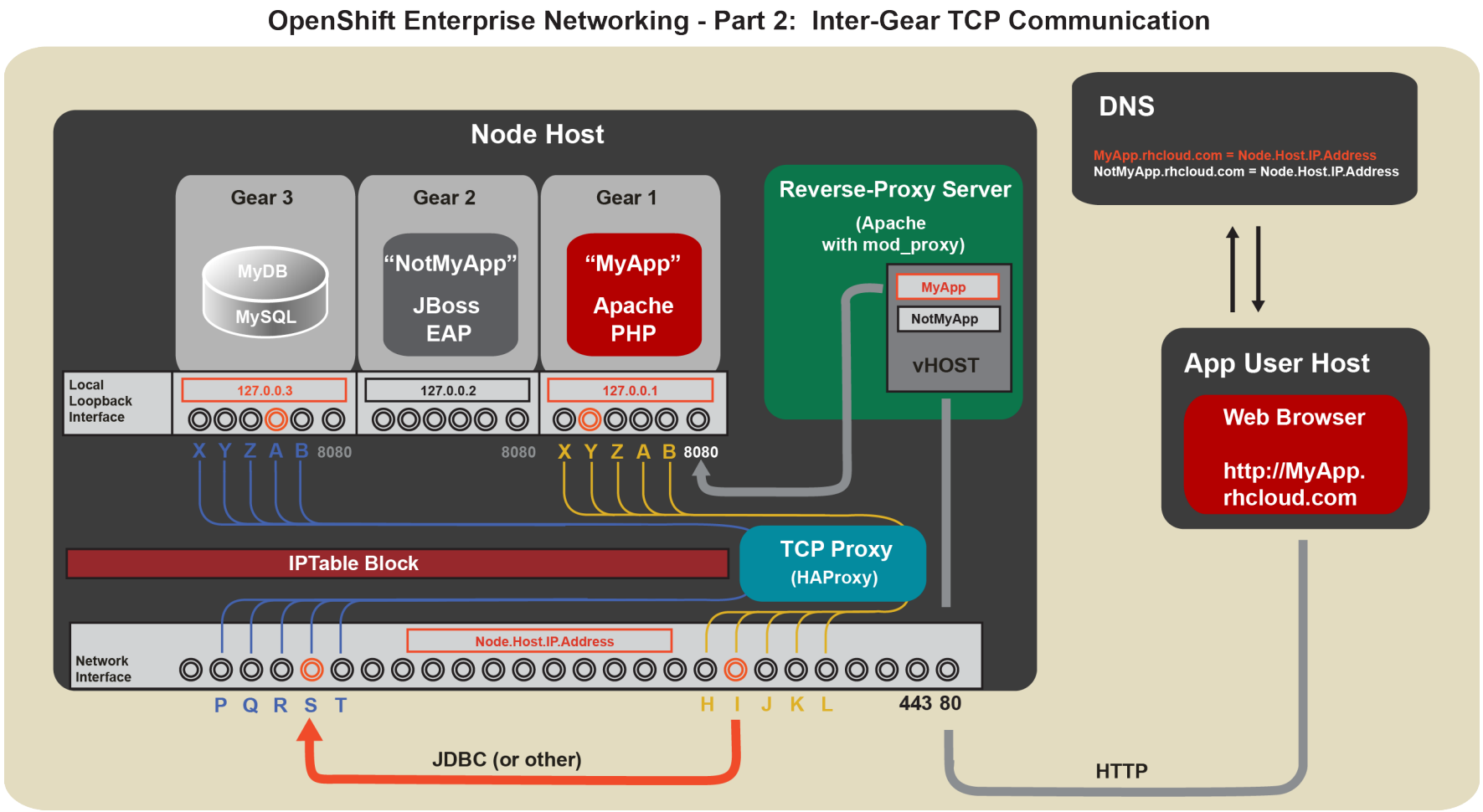
OpenShift Architecture



Application Communications – Part 1
– Incoming Requests to User Applications



Application Communications – Part 2
– Inter-Gear Communications




redhat.