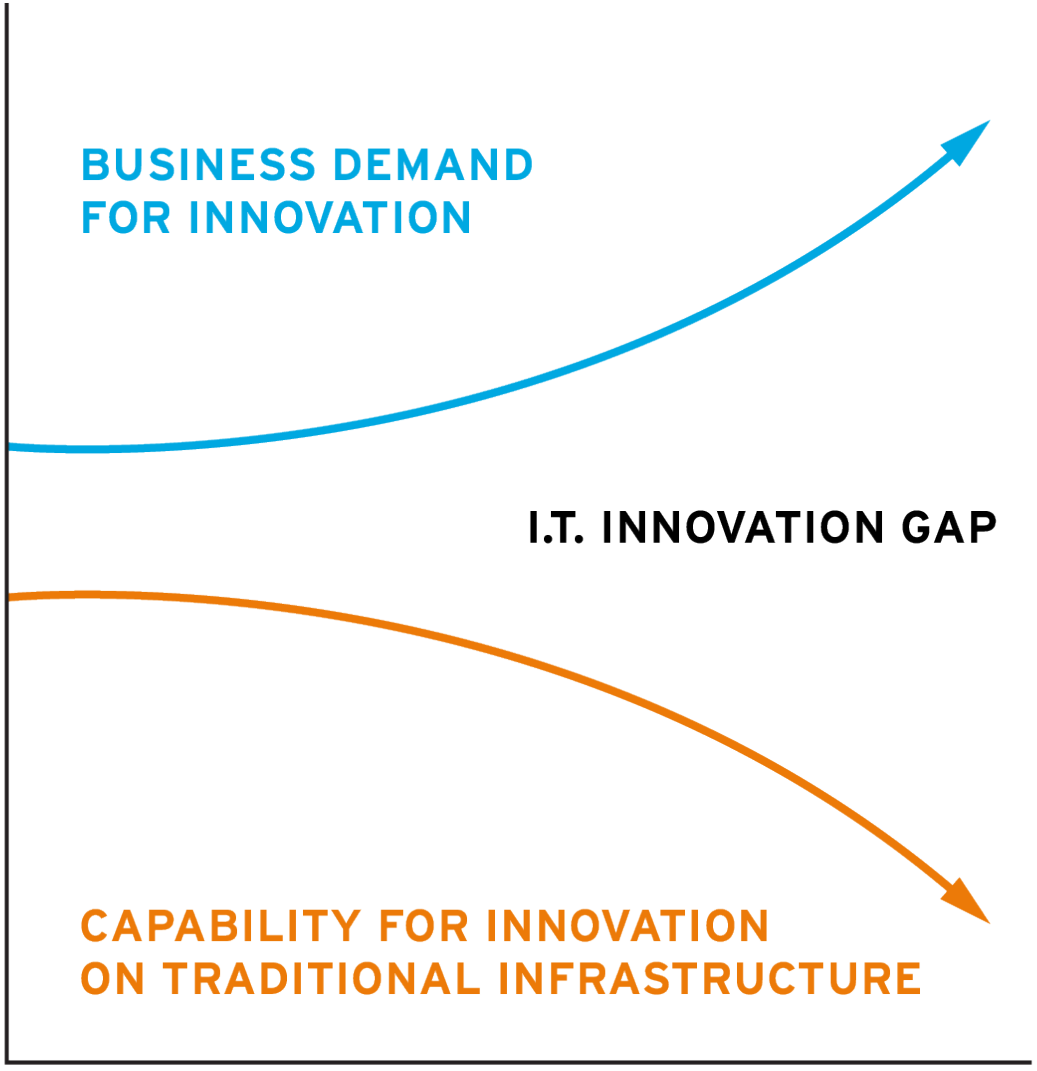
INTRODUCTION TO OPENSHIFT

Martin Sauvé
Architecte de solutions
21 janvier 2015

BUSINESS DEMANDS DRIVE I.T. TRANSFORMATION



- Business wants agility, lower cost, new capabilities
- IT struggling with existing legacy infrastructure architecture and cost model
- Cloud providers are using next-generation IT built on open source technologies
- IT needs to adopt cloud architectures and technologies to close innovation gap

I.T. TRANSFORMATION FOR EVERYONE

BUSINESS CHALLENGES

- Provide ubiquitous access to data and services
- Achieve better quality of service
- Rapid innovation and faster time to market

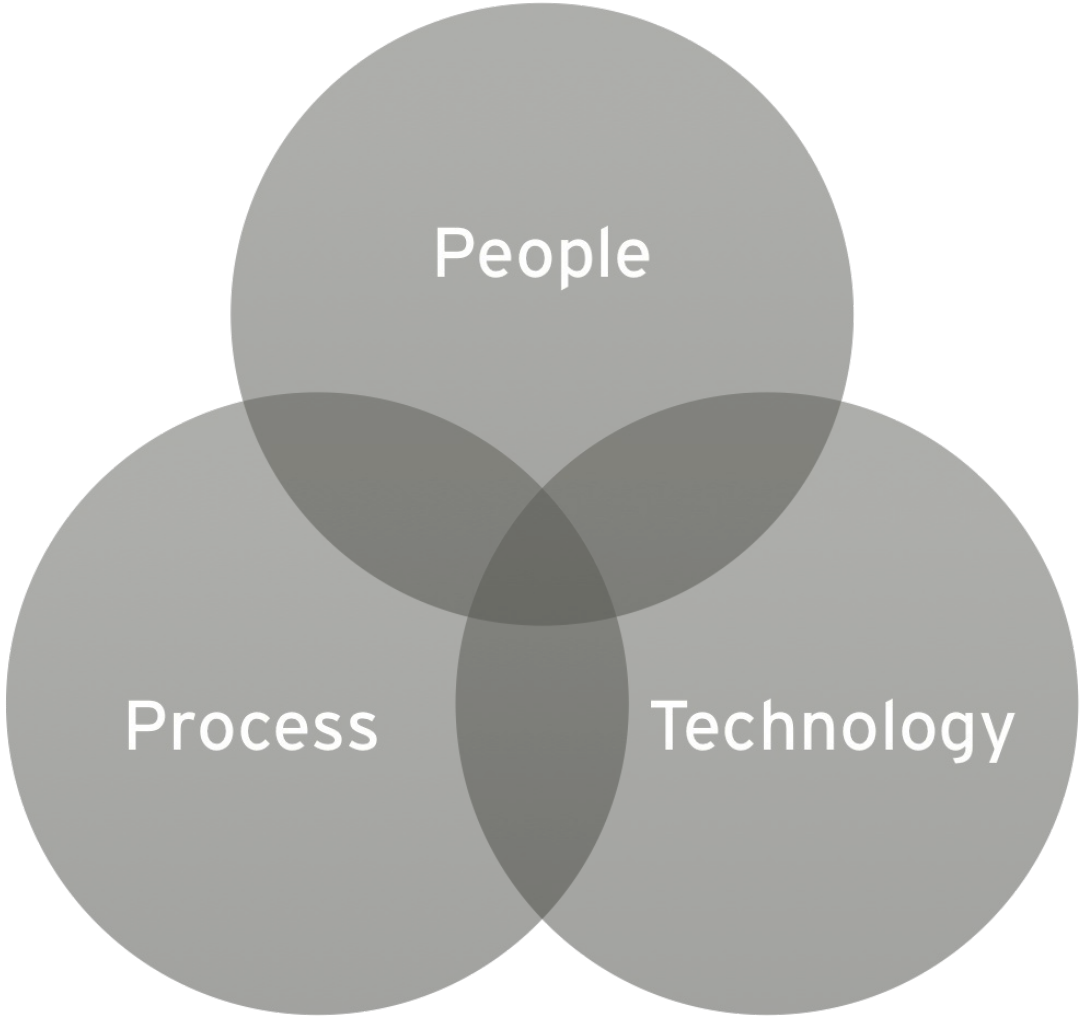
DEVELOPER CHALLENGES

- Reduce time to provision and develop, improve productivity
- Test new features and update applications faster
- Improve availability of platforms and resources

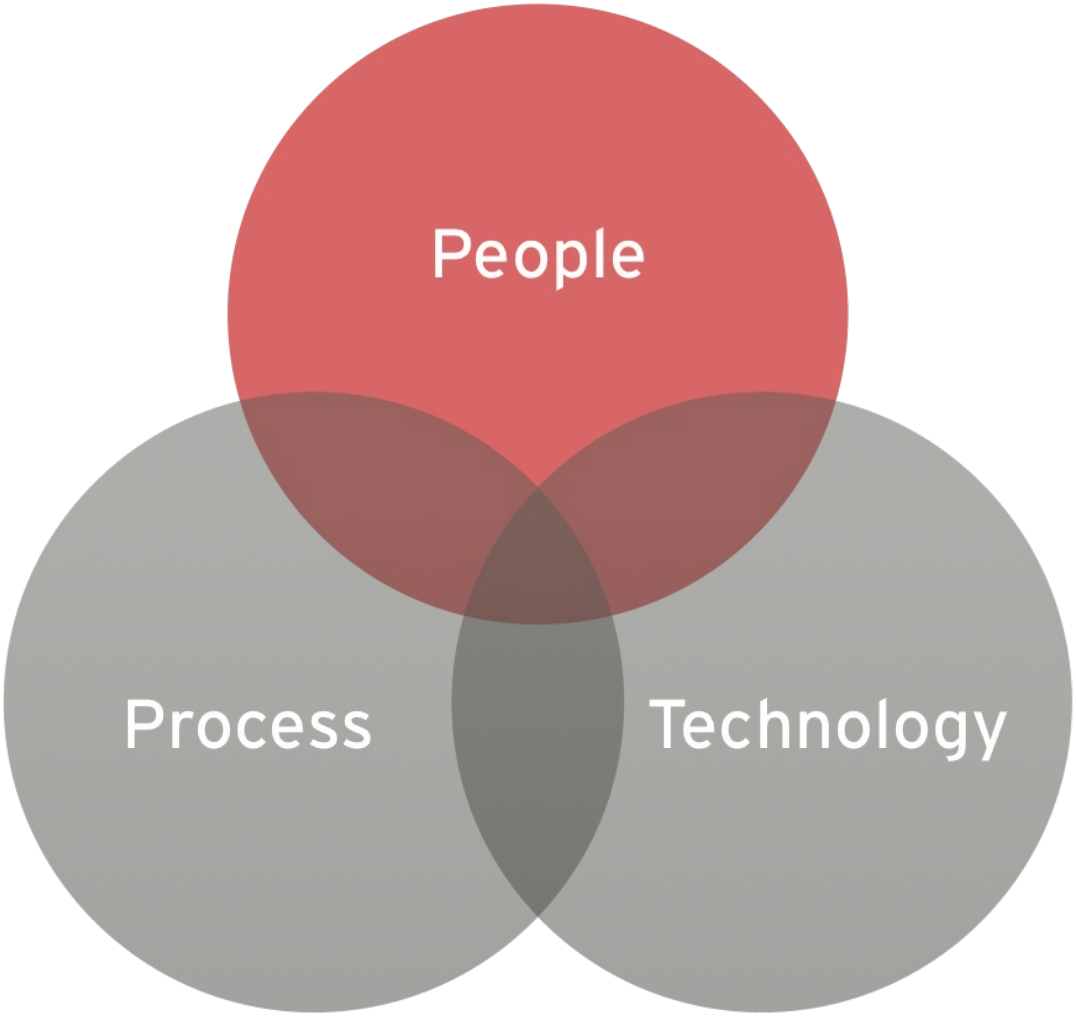
I.T. OPERATIONS CHALLENGES

- Increase operation efficiency
- Maximize resource utilization
- Reliable, secure, compliant

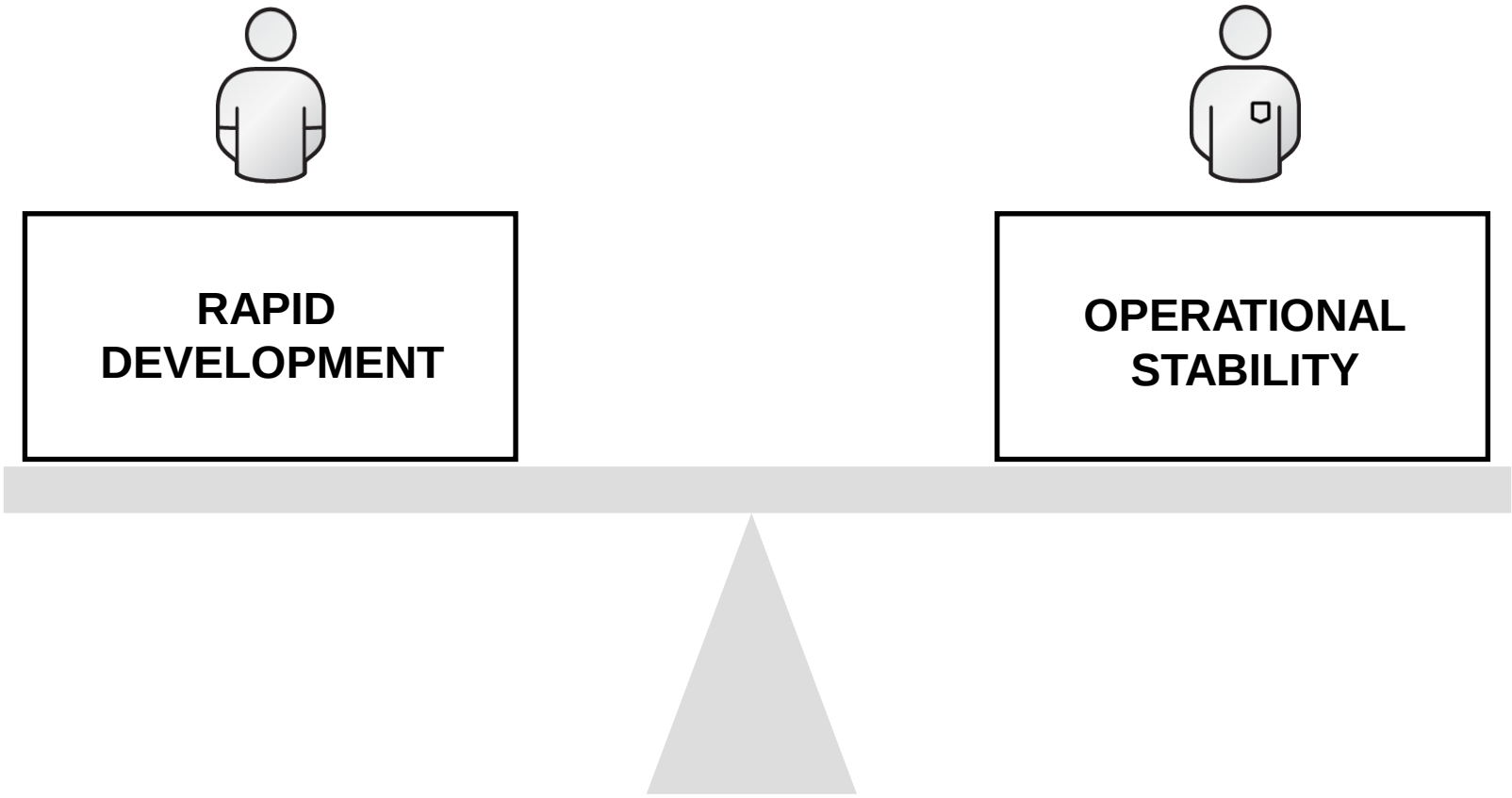
THREE PILLARS OF AN I.T. ORGANIZATION



THE PEOPLE

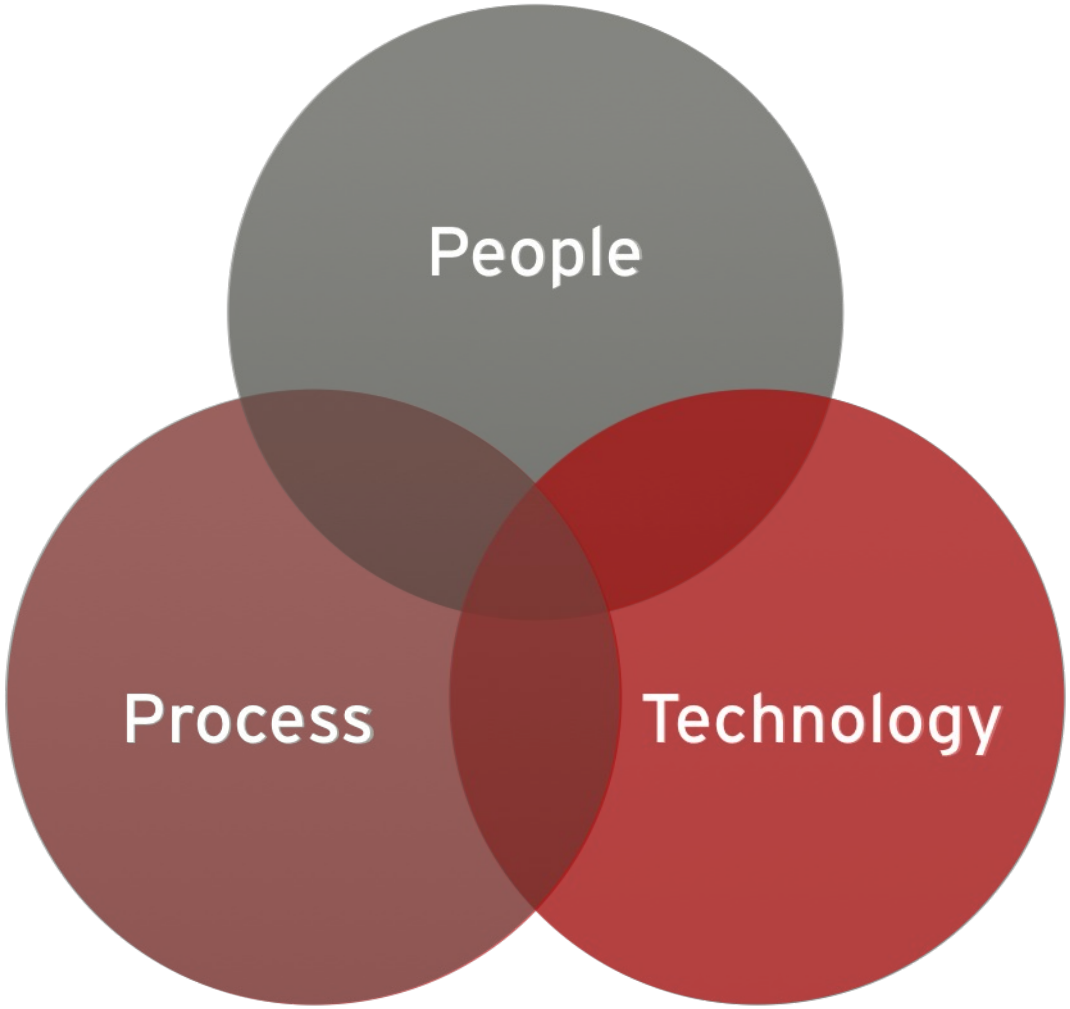


THE I.T. BALANCING ACT



To meet the growing demands of the business, developers and I.T. operations must find balance...

THE PROCESS AND THE TECHNOLOGY



TYPICAL DEVELOPMENT LIFECYCLE

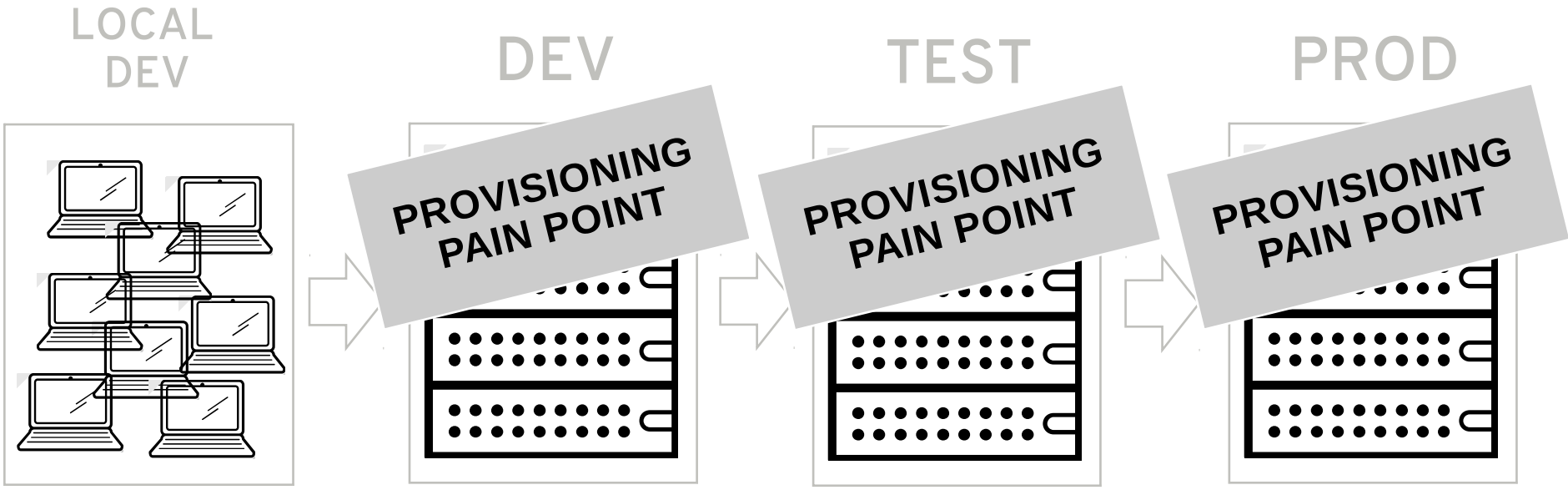
PHYSICAL

1. Have Idea
2. Get Budget
3. Submit Hardware Request
4. Wait…
5. Get Hardware
6. Rack and Stack Hardware
7. Install Operating System
8. Install Operating System Patches
9. Create User Accounts
10. Deploy Application Server
11. Deploy Framework/Tools
12. Code
13. Test
14. Buy and Configure Prod Servers
15. Push to Prod
16. Launch
17. Order More Servers to Meet Demand
18. Wait…
19. Deploy New Servers
20. Etc.

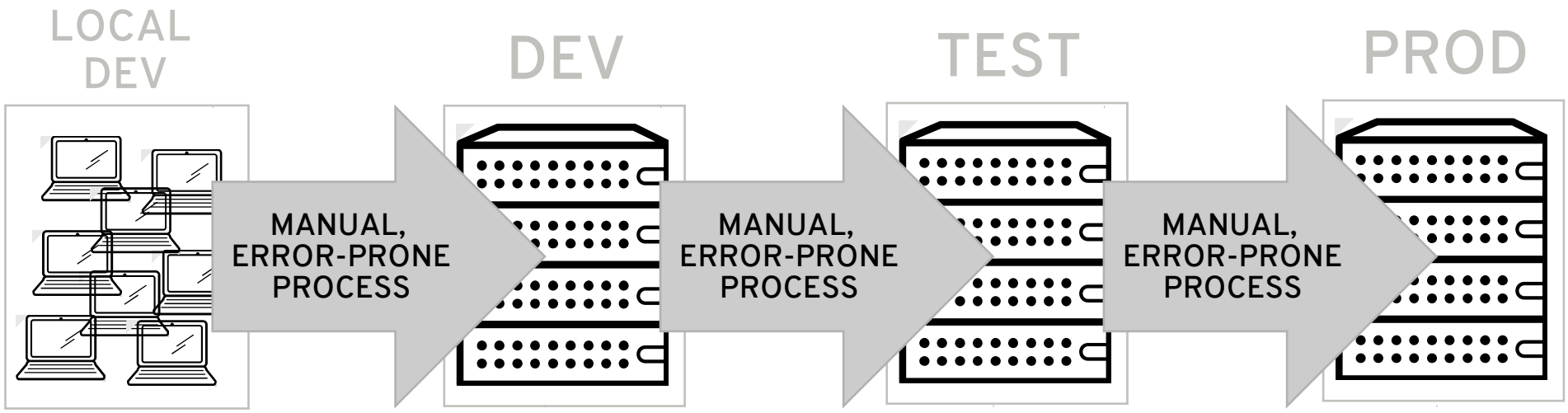
VIRTUAL

1. Have Idea
2. Get Budget
3. Submit VM Request
4. Wait…
5. Deploy Application Server
6. Deploy Framework/Tools
7. Code
8. Test
9. Configure Prod VMs
10. Push to Prod
11. Launch
12. Request VMs to Meet Demand
13. Wait…
14. Deploy New VMs
15. Etc.

TYPICAL ENVIRONMENT PROVISIONING



TYPICAL DEPLOYMENT PIPELINE



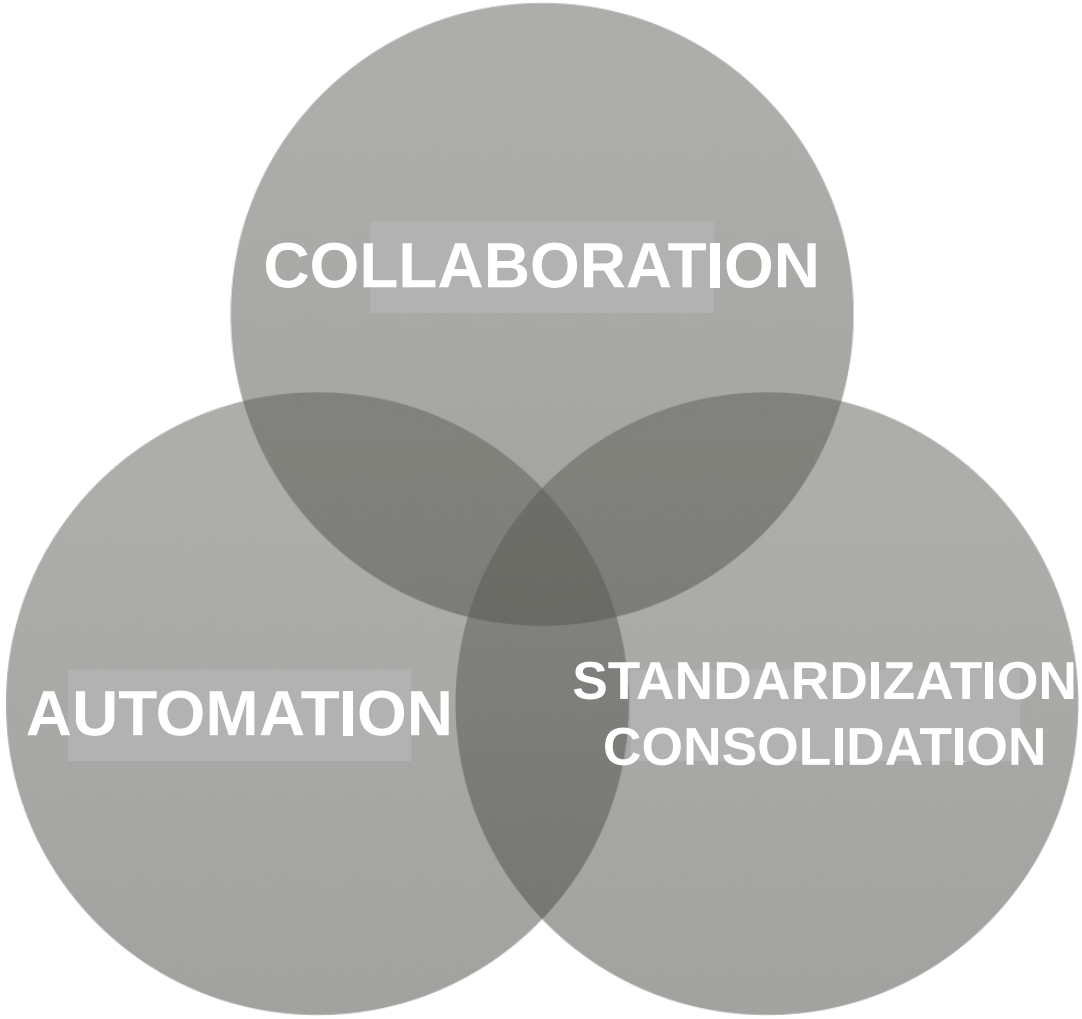
WHAT IF...

We could **automate** environment provisioning?

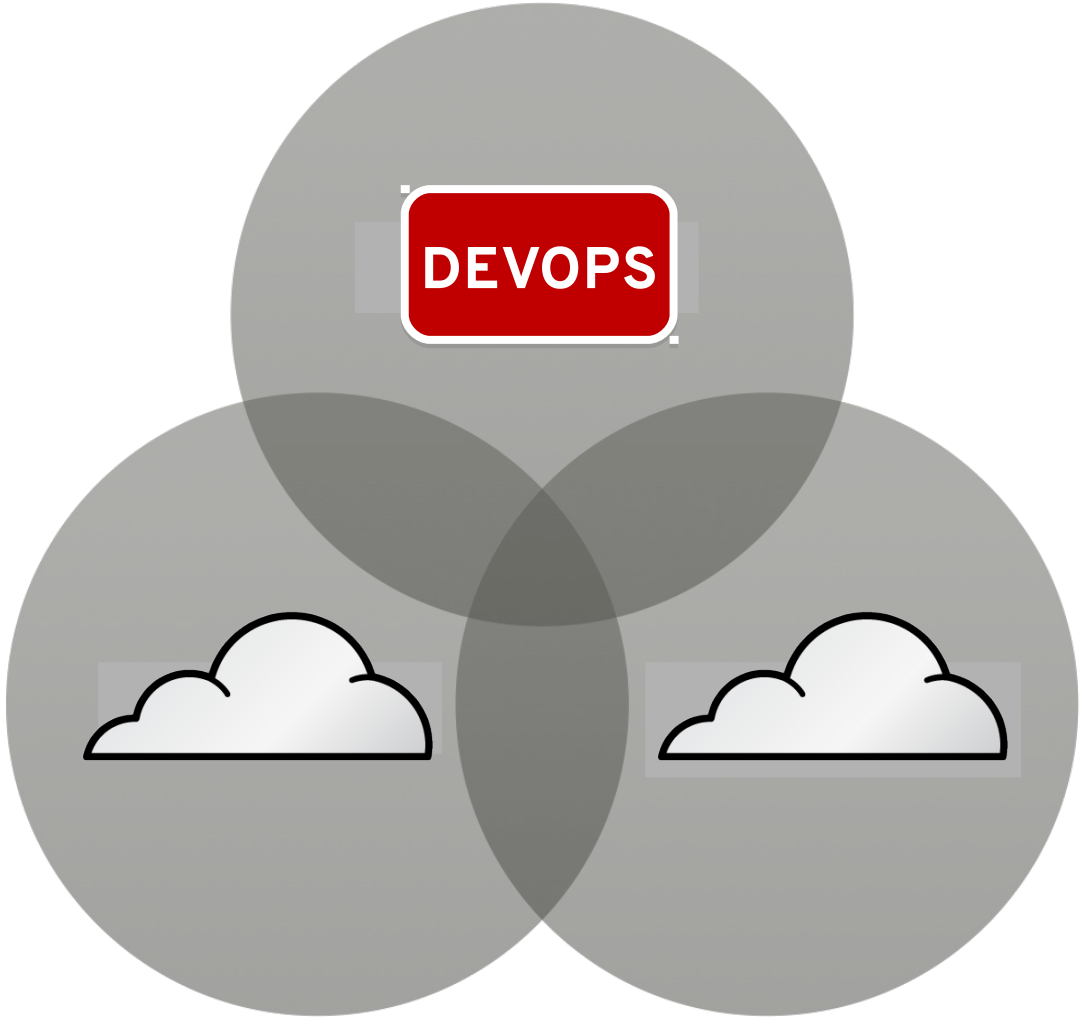
We could **standardize** technology stacks and platforms?

We could **consolidate** our resources and pool usage?

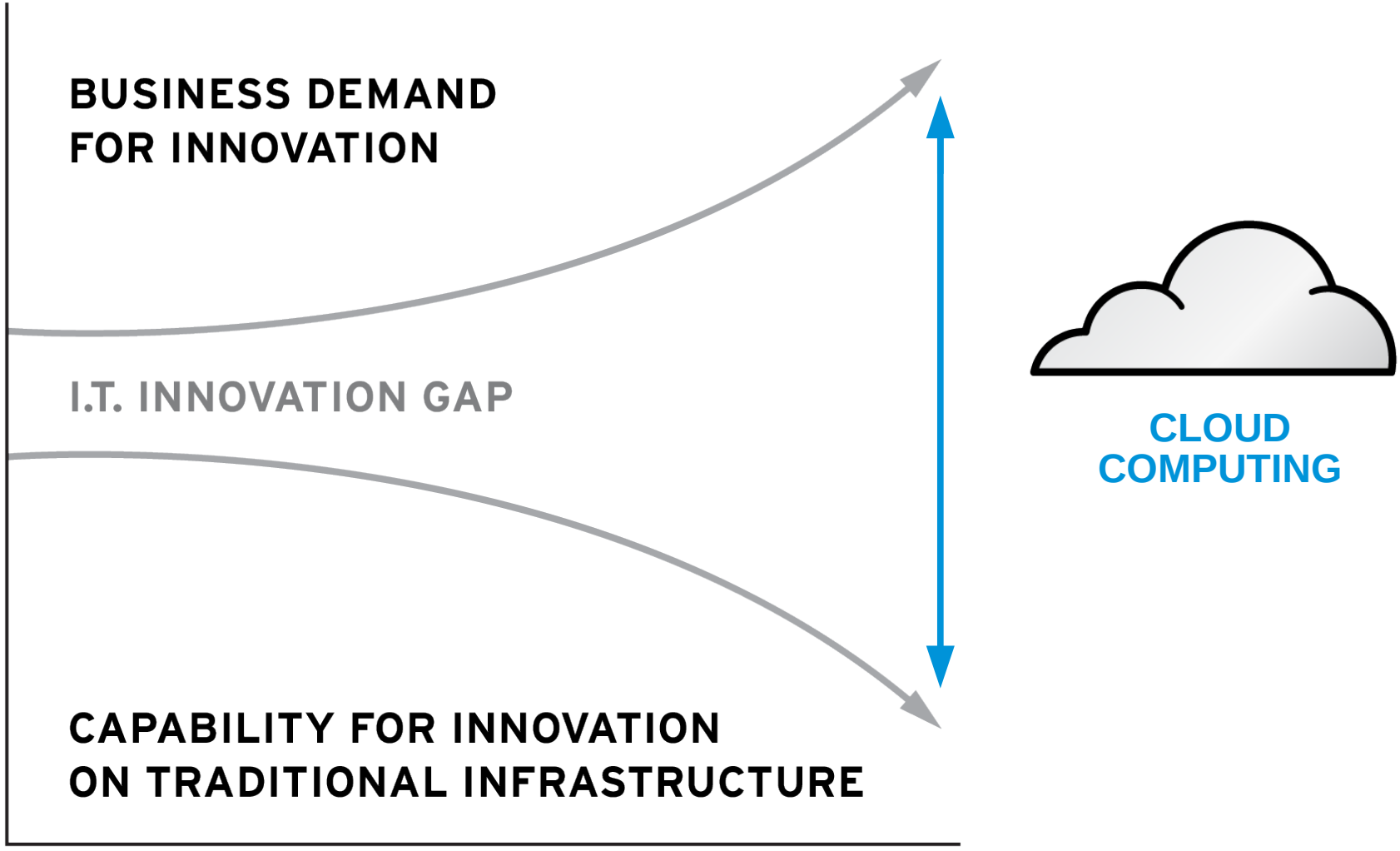
REALIZING I.T. EFFICIENCY



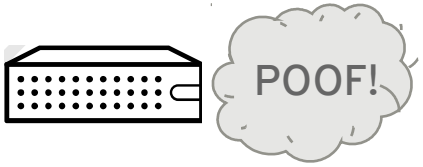
APPLYING THE METHODOLOGIES



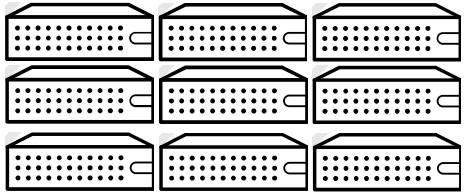
CLOUD CLOSES THE INNOVATION GAP



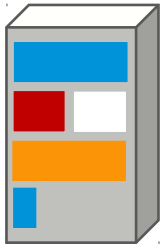
CLOUD SERVERS ARE...



EPHEMERAL



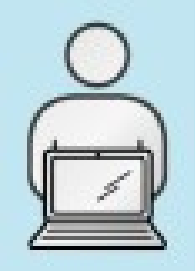
ANONYMOUS



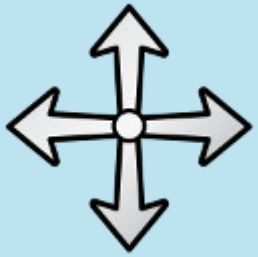
MULTI-TENANT

CLOUD COMPUTING CHARACTERISTICS

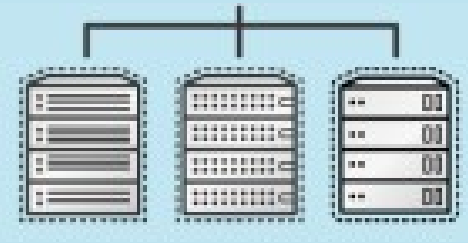
SELF-SERVICE PROVISIONING



RAPID ELASTICITY

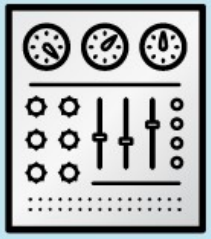


RESOURCE POOLING

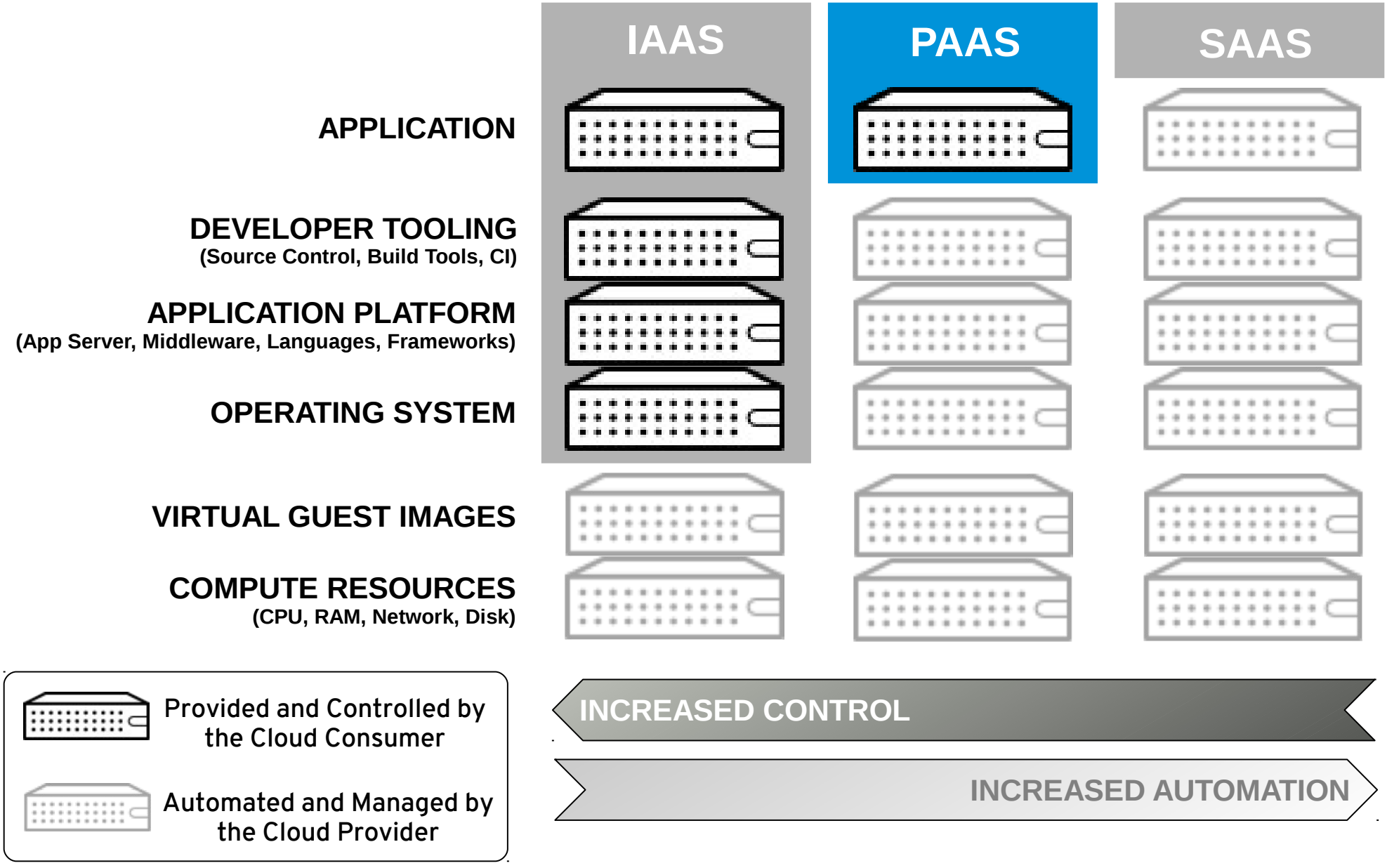


NETWORK ACCESS

MEASURED SERVICE



CLOUD SERVICE MODELS

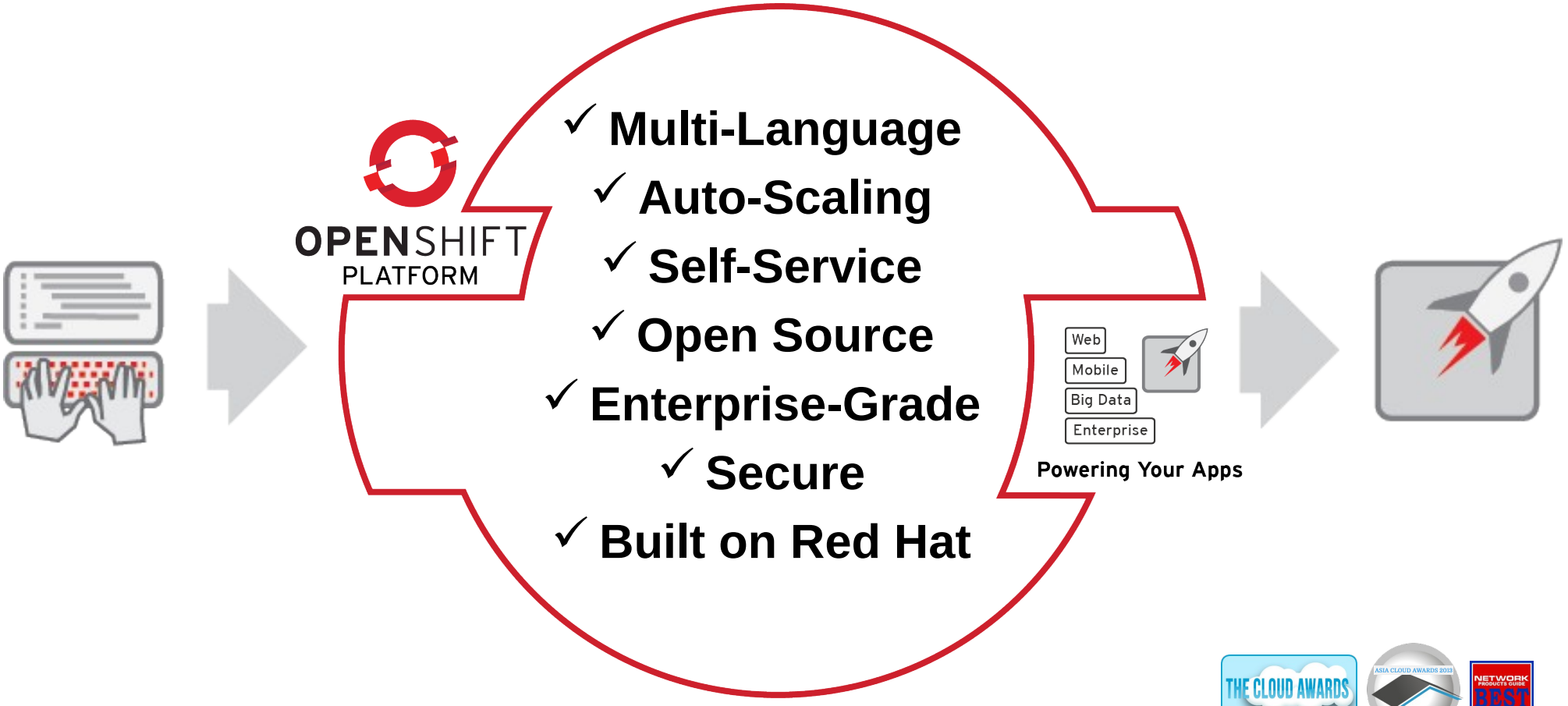


IMPLEMENTING A PAAS

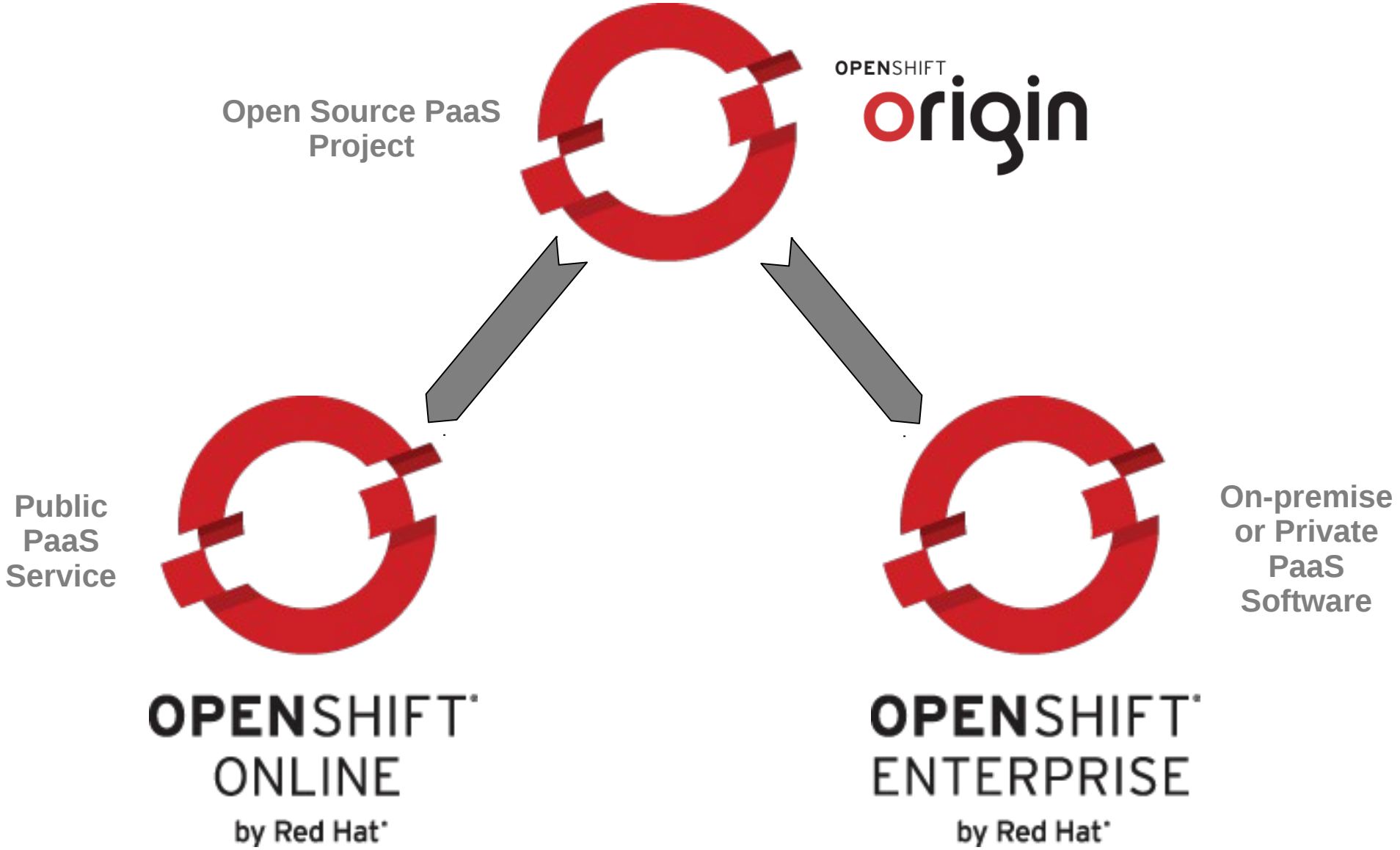
Gartner

> "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 IS PAAS BY RED HAT

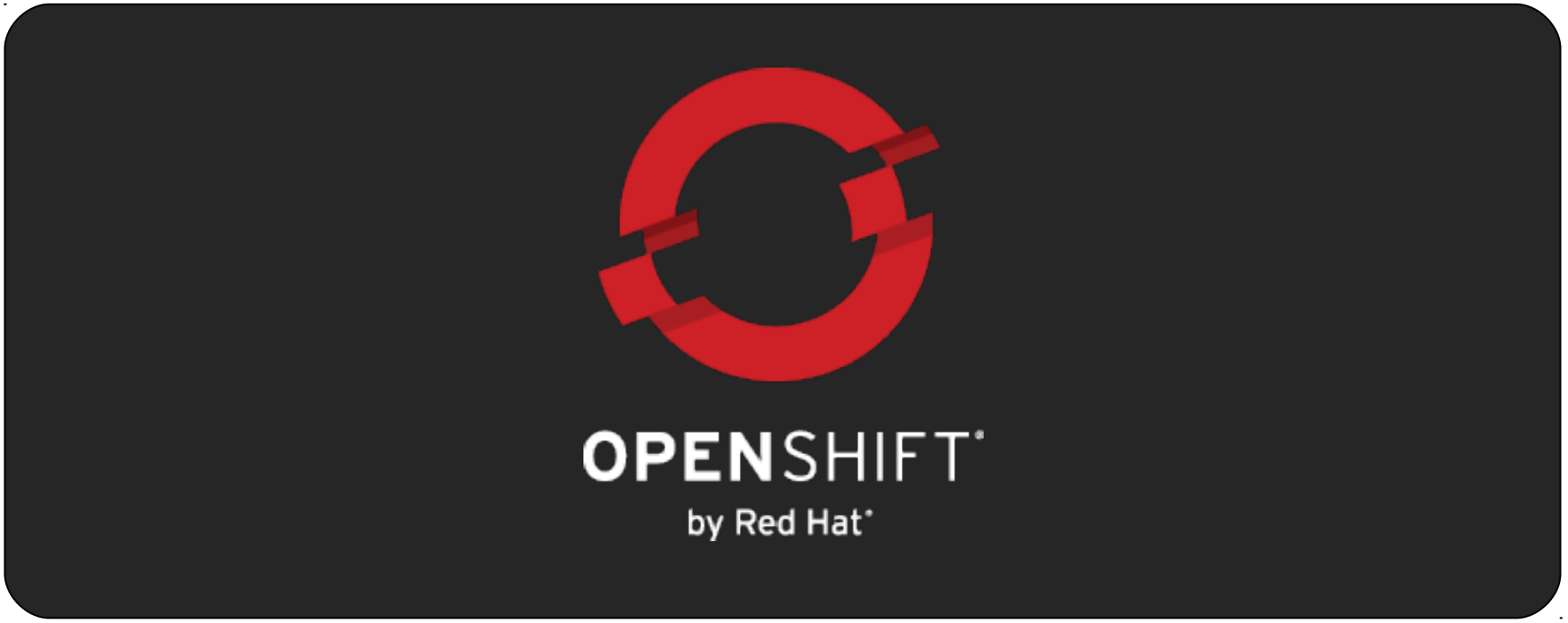


RED HAT'S PAAS STRATEGY



OPENSHIFT by Red Hat

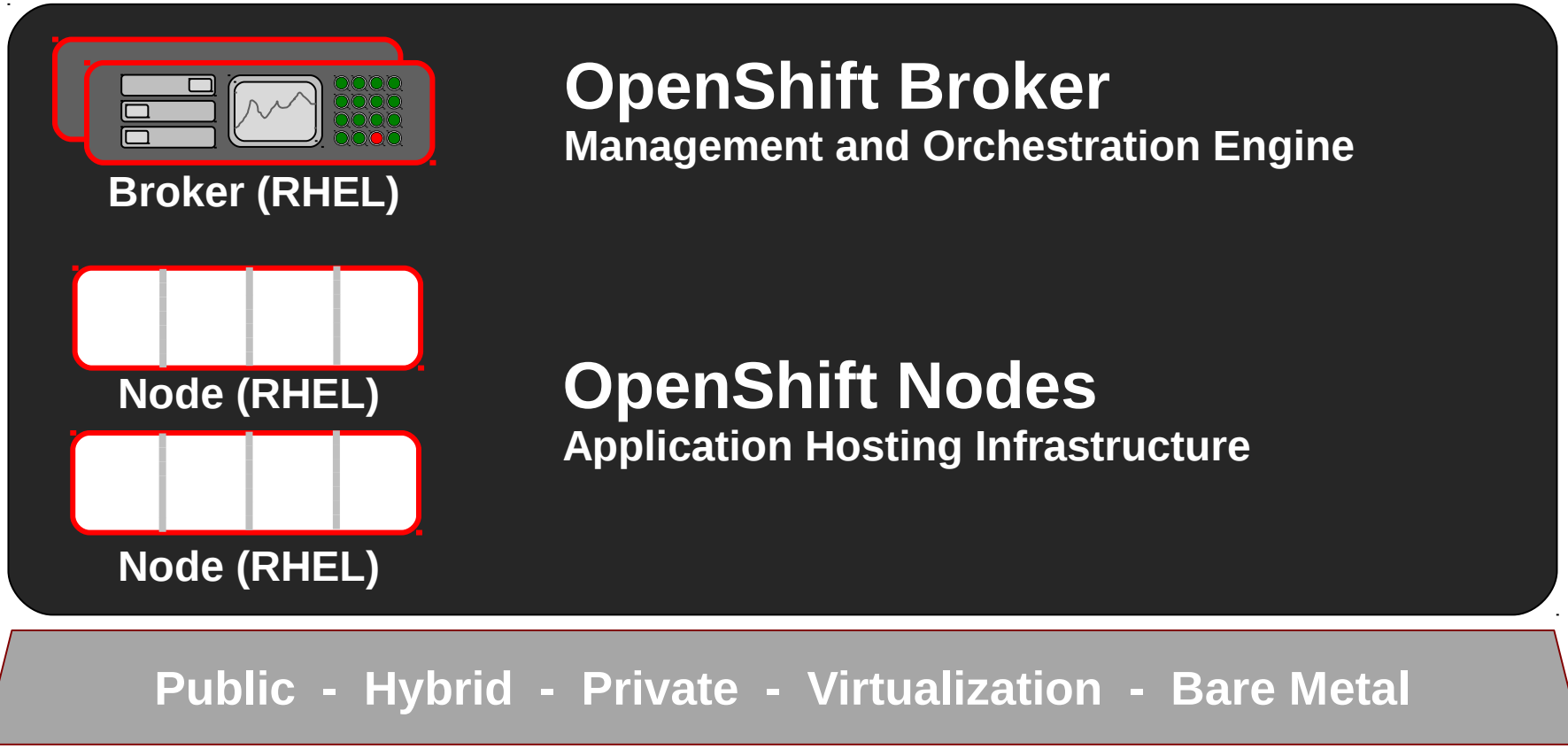
OPENSHIFT PAAS ON YOUR CHOICE OF CLOUD OR INFRASTRUCTURE...



Public - Hybrid - Private - Virtualization - Bare Metal

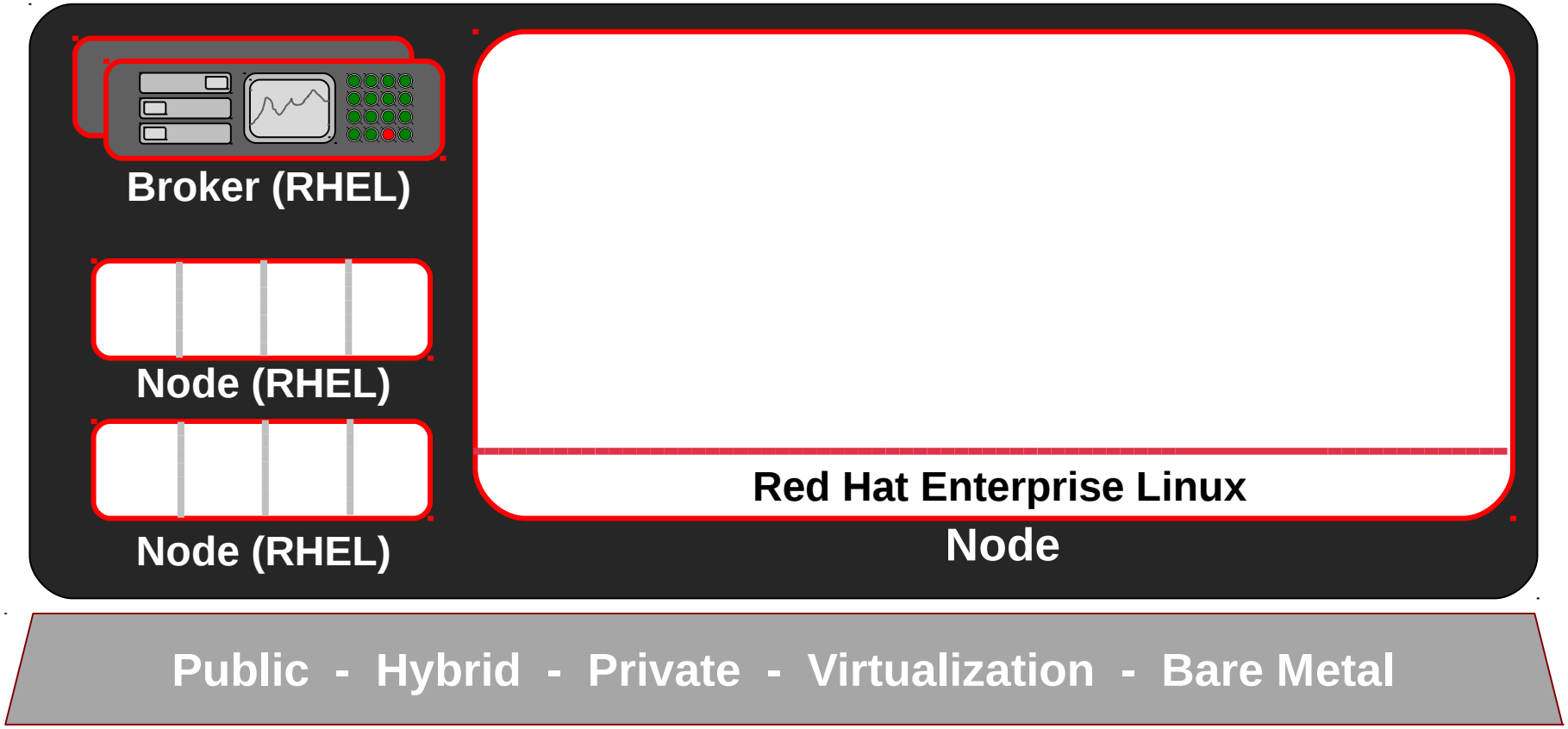
OPENSHIFT by Red Hat

AN OPENSHIFT BROKER MANAGES MULTIPLE OPENSHIFT NODES



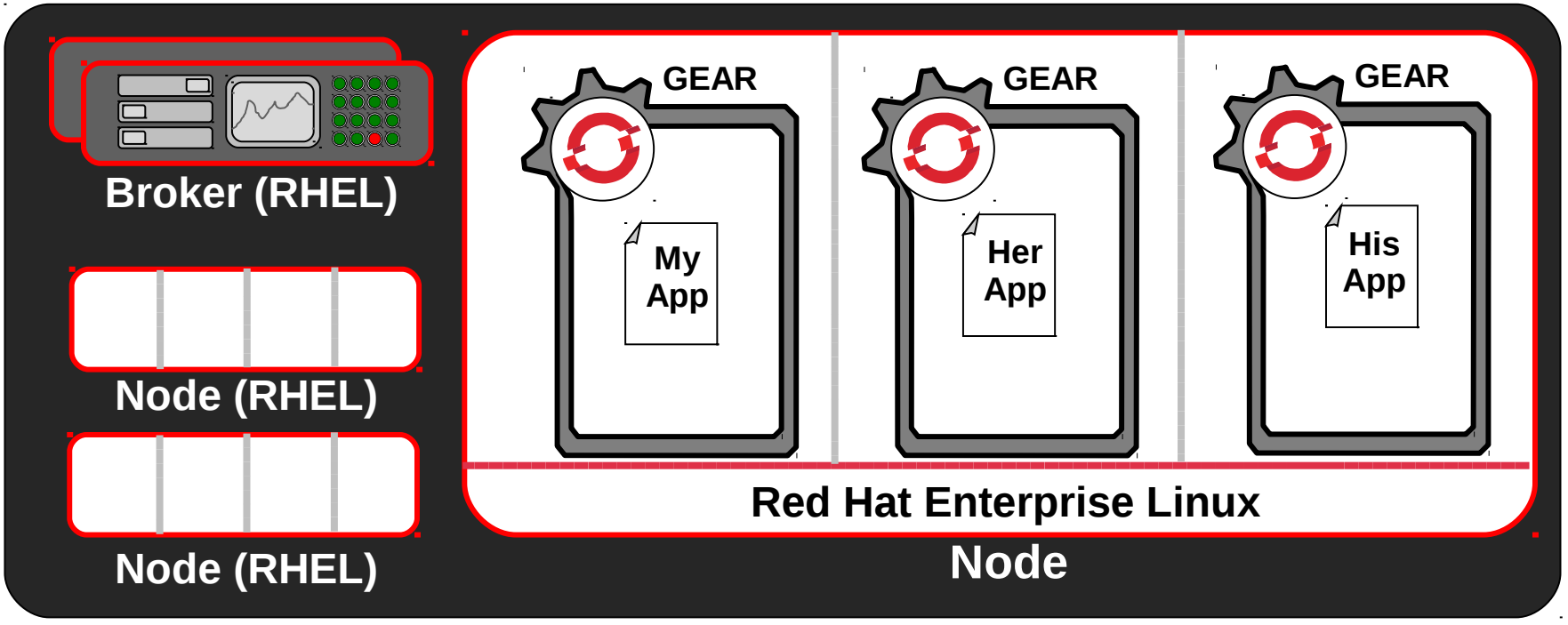
OPENSHIFT by Red Hat

A NODE IS AN INSTANCE OF RHEL



OPENSHIFT by Red Hat

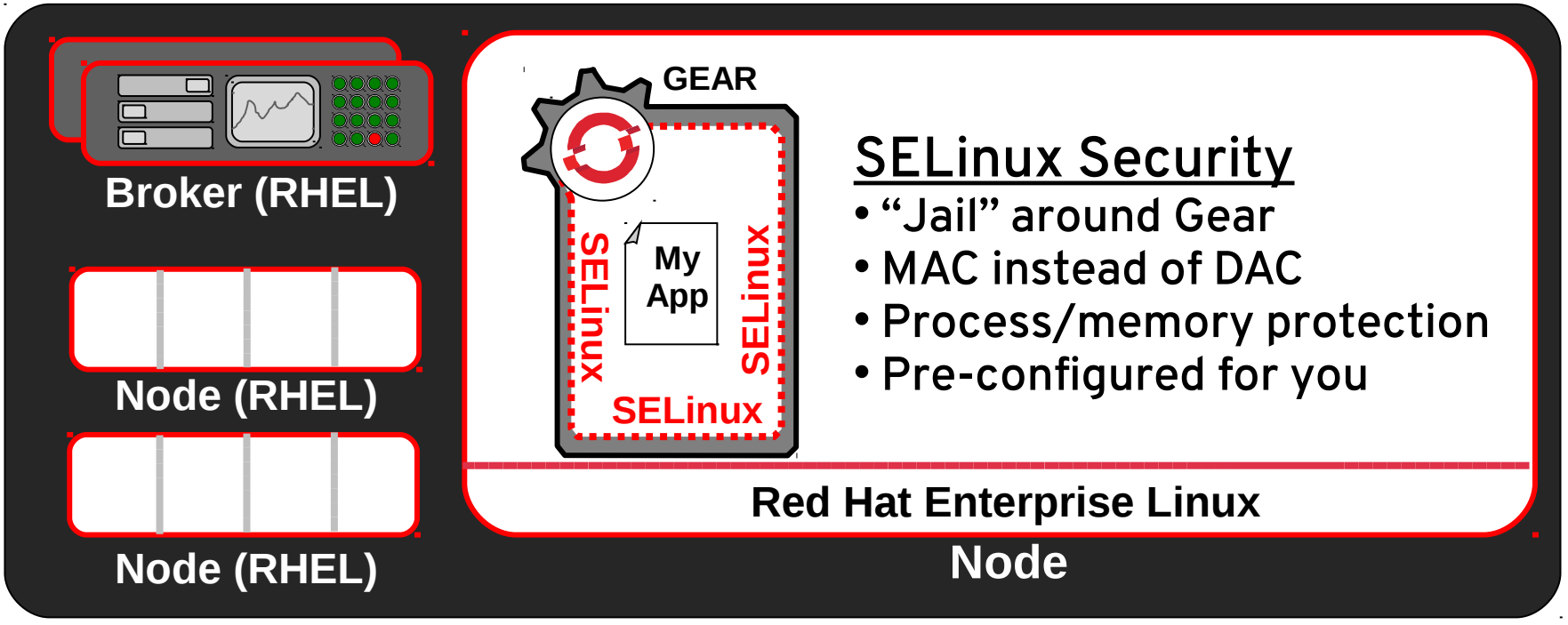
OPENSHIFT USER APPLICATIONS RUNS IN CONTAINERS CALLED GEARS



redhat.

OPENSHIFT by Red Hat

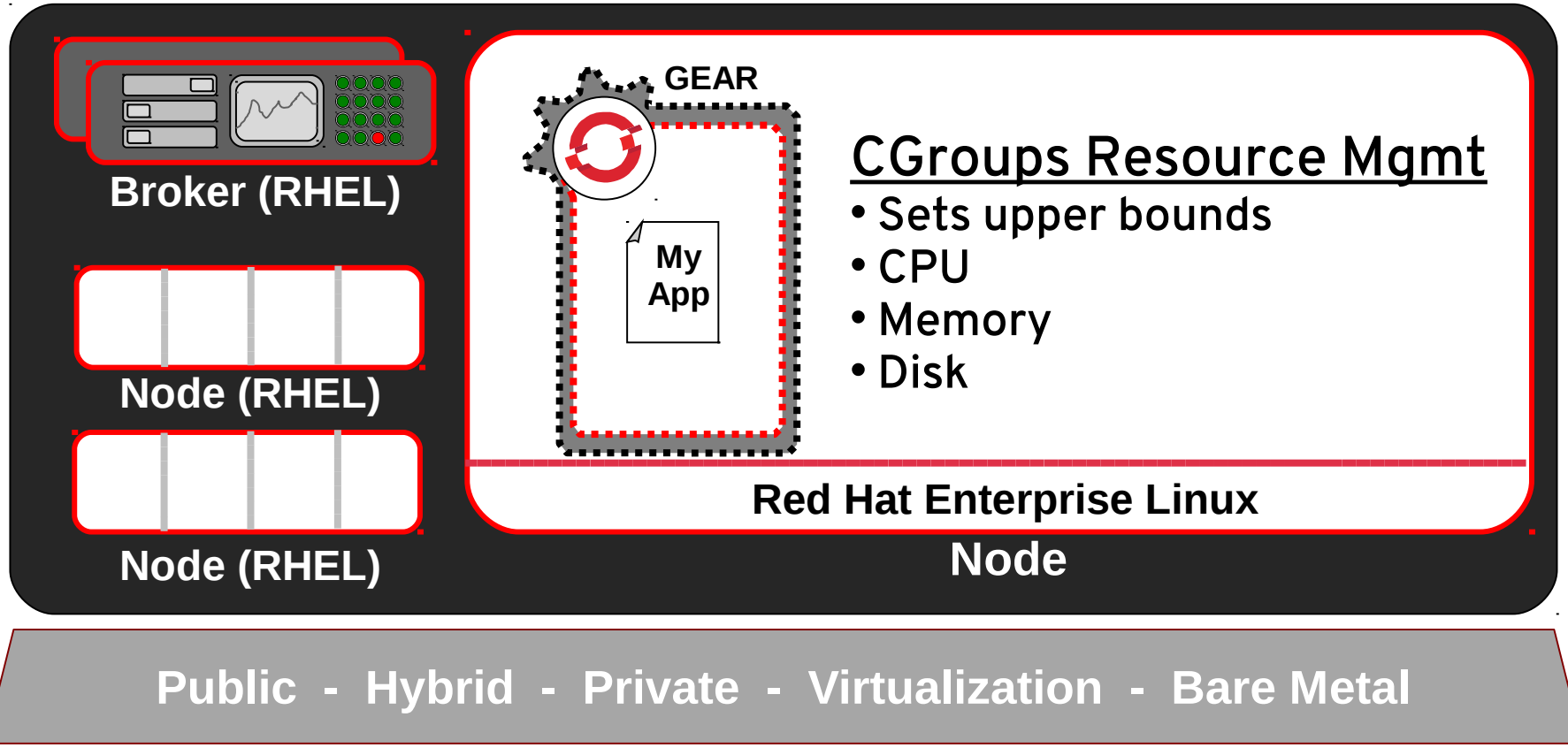
GEARS USE SELINUX FOR PRE-CONFIGURED SECURITY



redhat.

OPENSHIFT by Red Hat

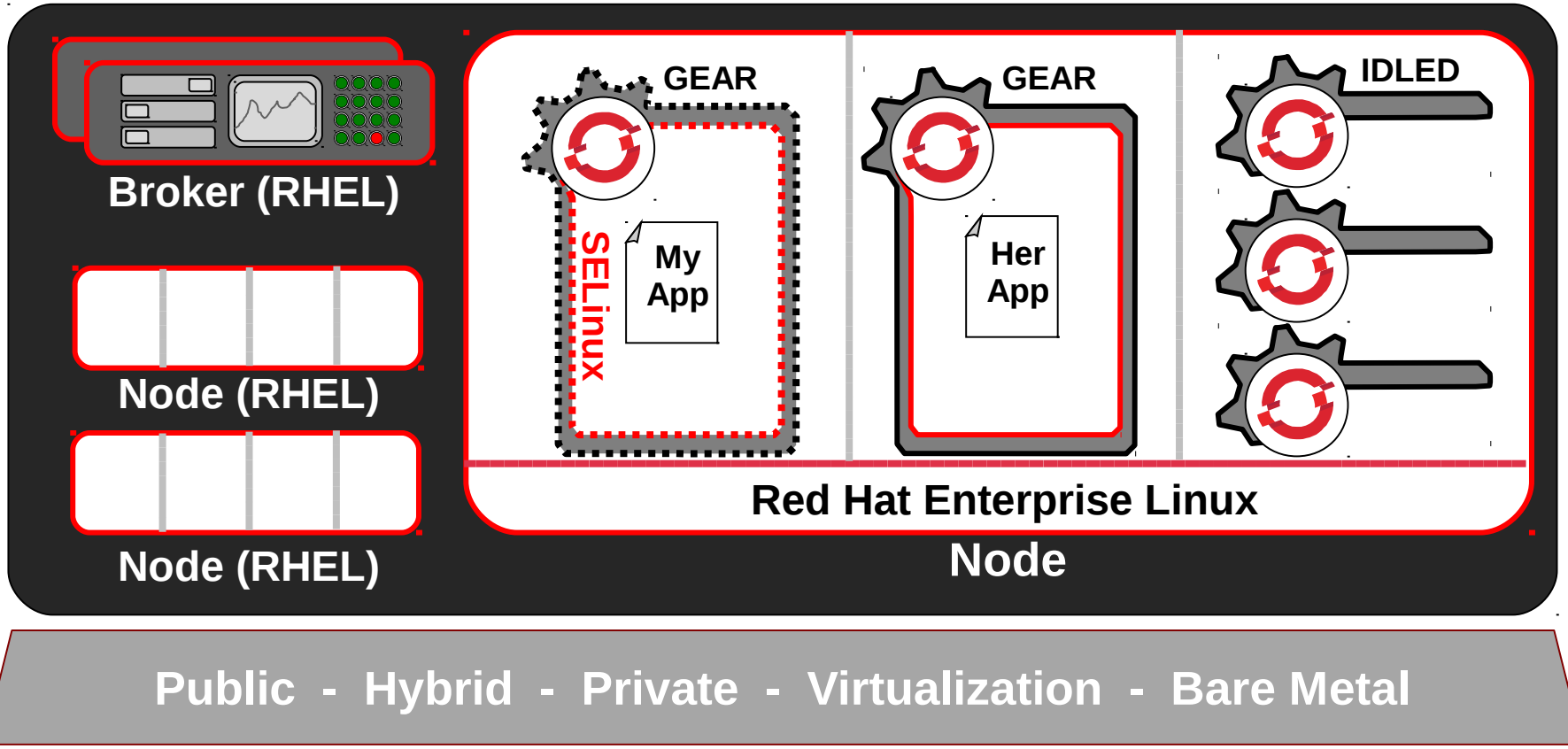
GEARS USE LINUX CGROUPS FOR RESOURCE MANAGEMENT



redhat.

OPENSHIFT by Red Hat

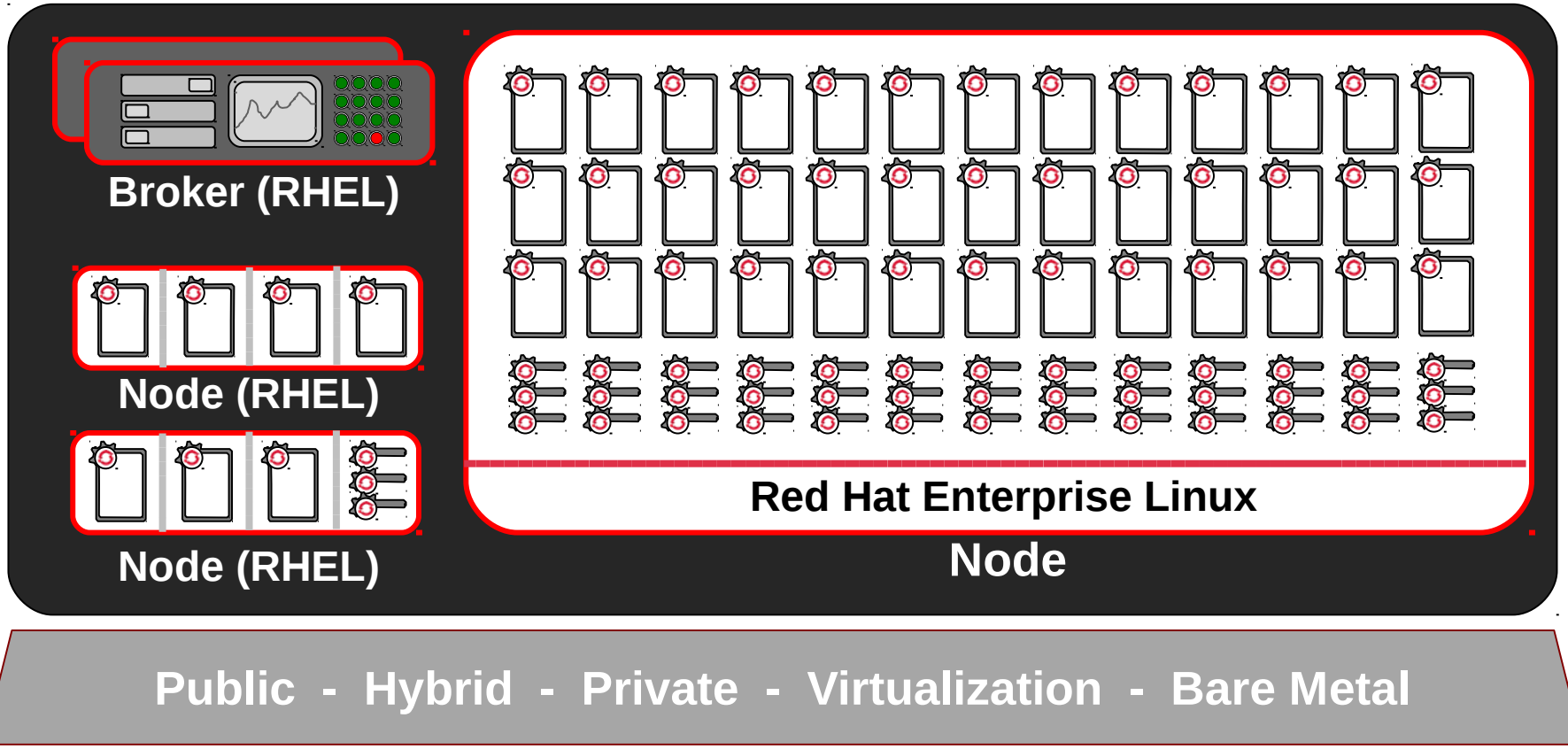
IDLE GEARS CAN BE “DE-HYDRATED” BY THE OPENSHIFT BROKER



redhat.

OPENSHIFT by Red Hat

OPENSHIFT MULTI-TENANCY PROVIDES DENSITY, EFFICIENCY, AND SECURITY



OpenShift Nodes

- OpenShift Nodes are assigned a Node Profile
  - Node Profile describes resources (RAM, CPU...) available to each gear
- Nodes are assigned a capacity (max_gears)
- Nodes have a default gear quota by user
- Max gears can be adjusted for individual users
  - oo-admin-ctl-user -l username --setmaxgears 101

STREAMLINING DEVELOPMENT WITH OPENSHIFT

Gartner

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

TYPICAL DEVELOPMENT LIFECYCLE

PHYSICAL

1. Have Idea
2. Get Budget
3. Submit Hardware Request
4. Wait…
5. Get Hardware
6. Rack and Stack Hardware
7. Install Operating System
8. Install Operating System Patches
9. Create User Accounts
10. Deploy Application Server
11. Deploy Framework/Tools
12. Code
13. Test
14. Buy and Configure Prod Servers
15. Push to Prod
16. Launch
17. Order More Servers to Meet Demand
18. Wait…
19. Deploy New Servers
20. Etc.

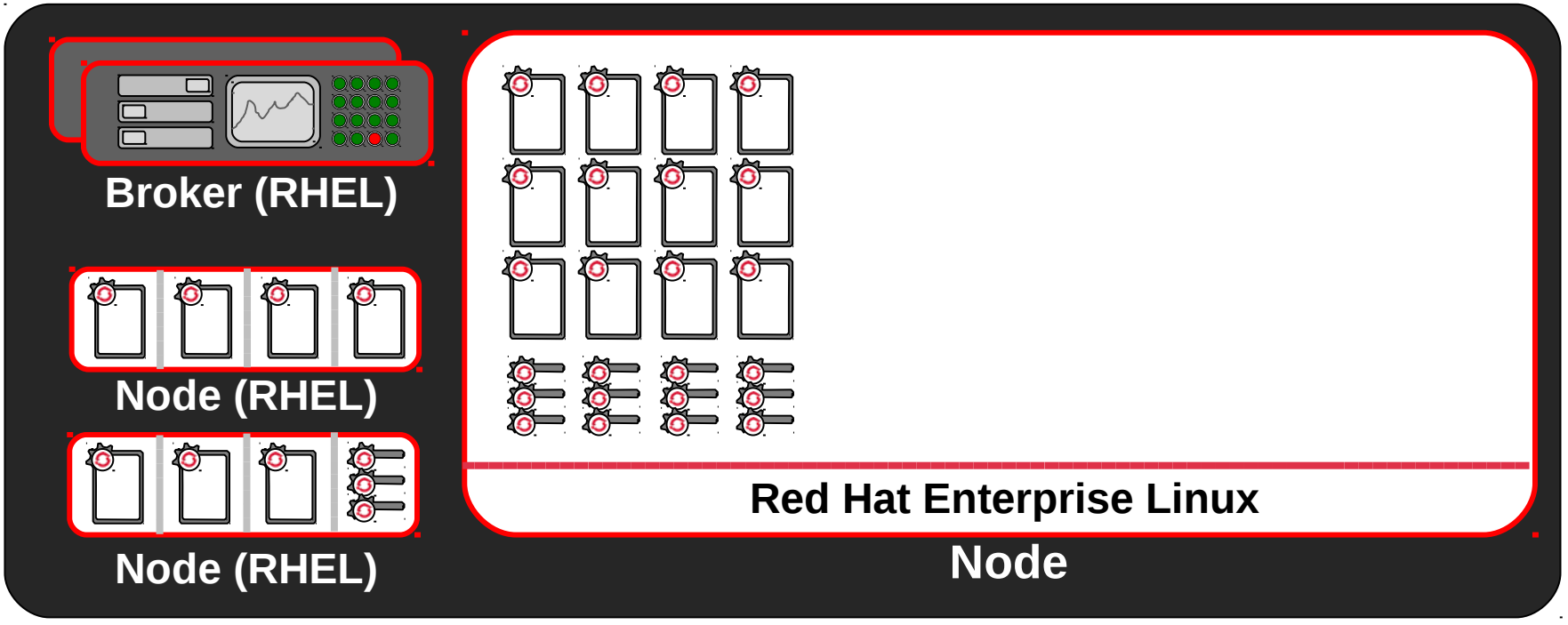
VIRTUAL

1. Have Idea
2. Get Budget
3. Submit VM Request
4. Wait…
5. Deploy Application Server
6. Deploy Framework/Tools
7. Code
8. Test
9. Configure Prod VMs
10. Push to Prod
11. Launch
12. Request VMs to Meet Demand
13. Wait…
14. Deploy New VMs
15. Etc.

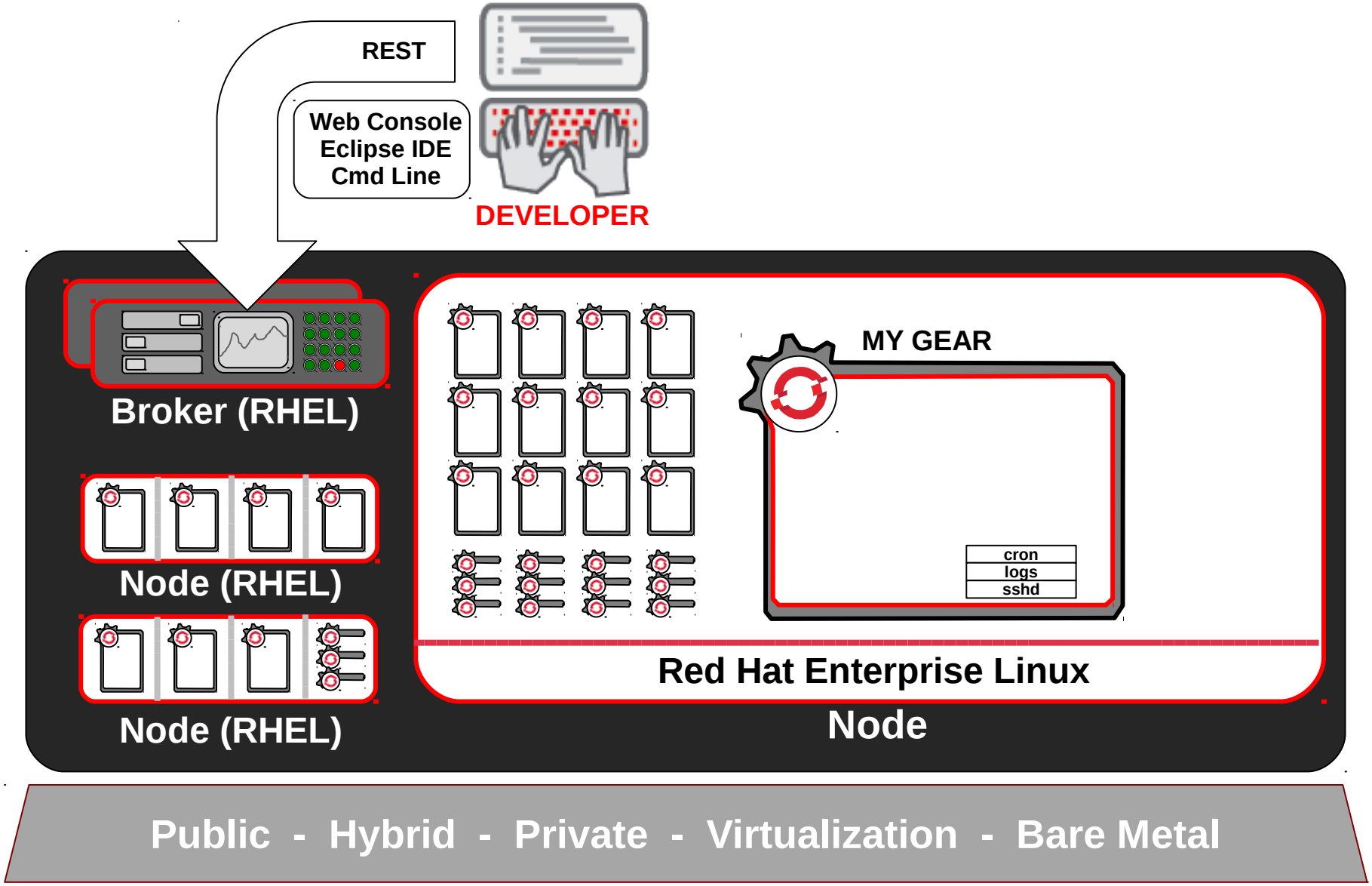
DEVELOPER WORKFLOW

DEVELOPER

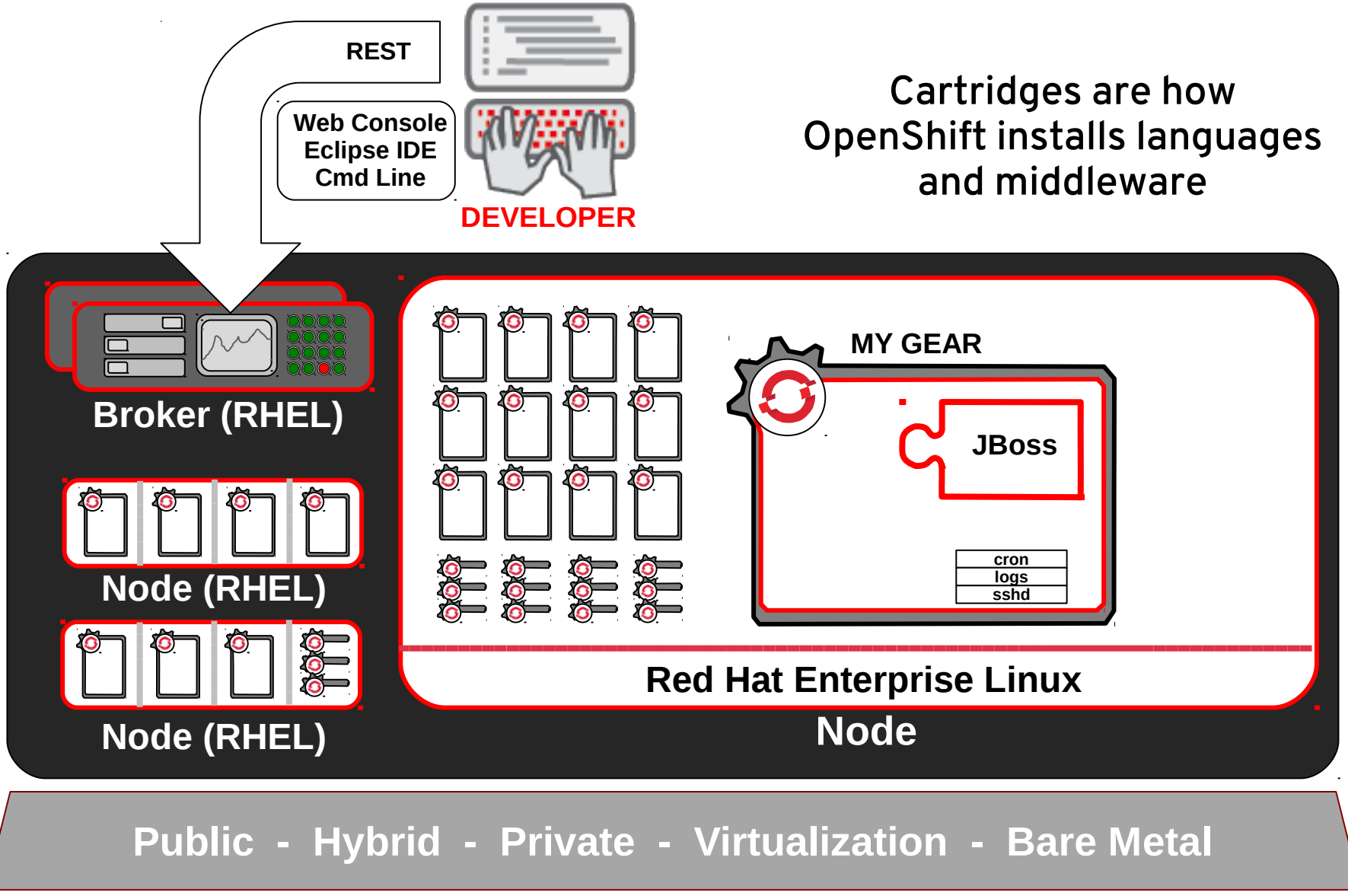
A developer has a new idea for an application. First, they need to create a new gear in OpenShift...



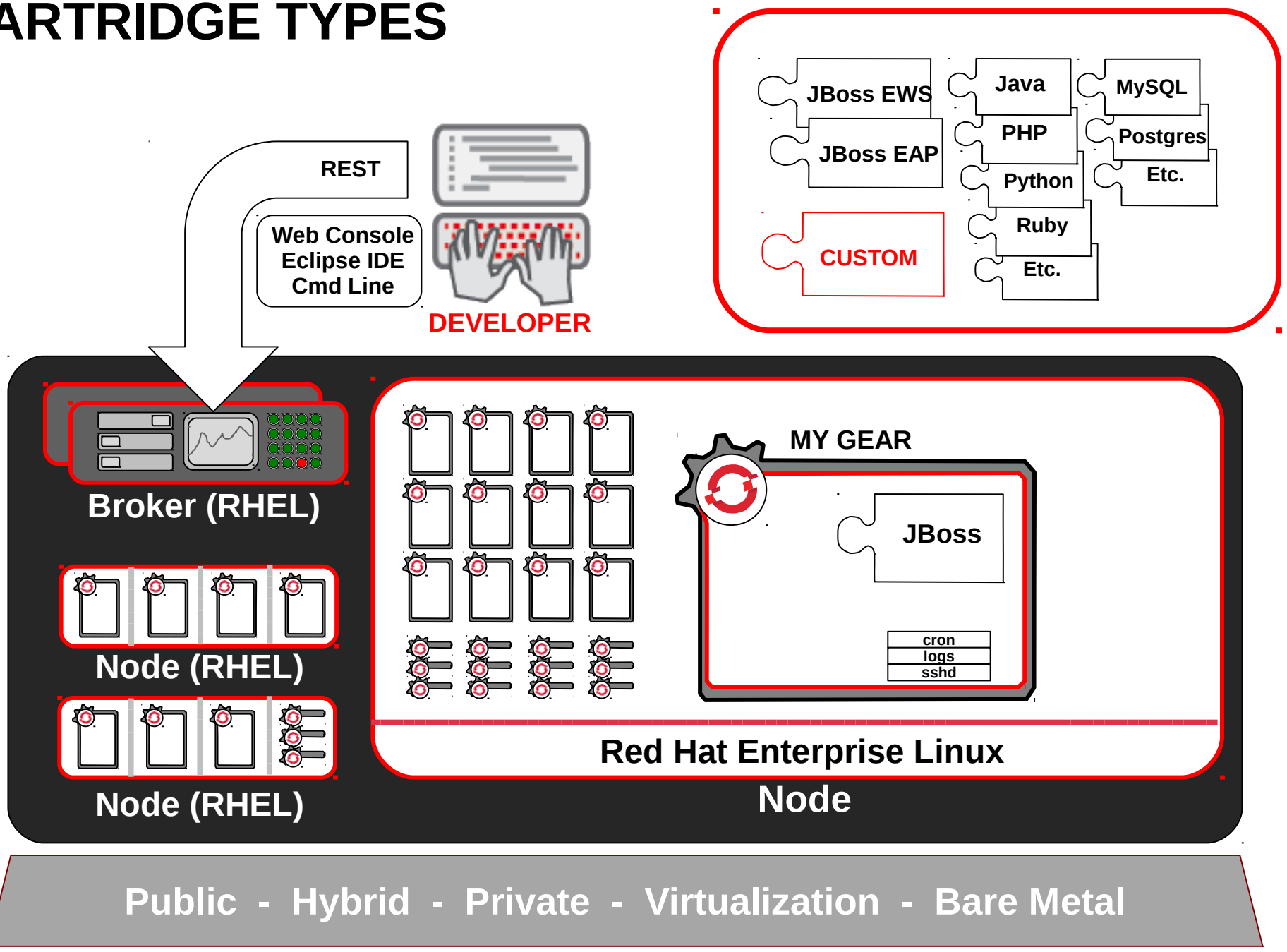
GEAR CREATION (WEB, CLI, ECLIPSE)



OPENSHIFT AUTOMATES GEAR CONFIGURATION VIA CARTRIDGES



CARTRIDGE TYPES



OpenShift Cartridges

OpenShift Enterprise cartridges are a mechanism to encapsulate application components, for example:

- language runtimes,
  - Java, PHP, Ruby, Python, Node.js, and Perl
- Middleware,
  - Jenkins CI server and client, Cron, and the HAProxy load balancer
- and databases
  - MySQL, PostgreSQL,

- expose them as services
  - so developers can use them to create applications

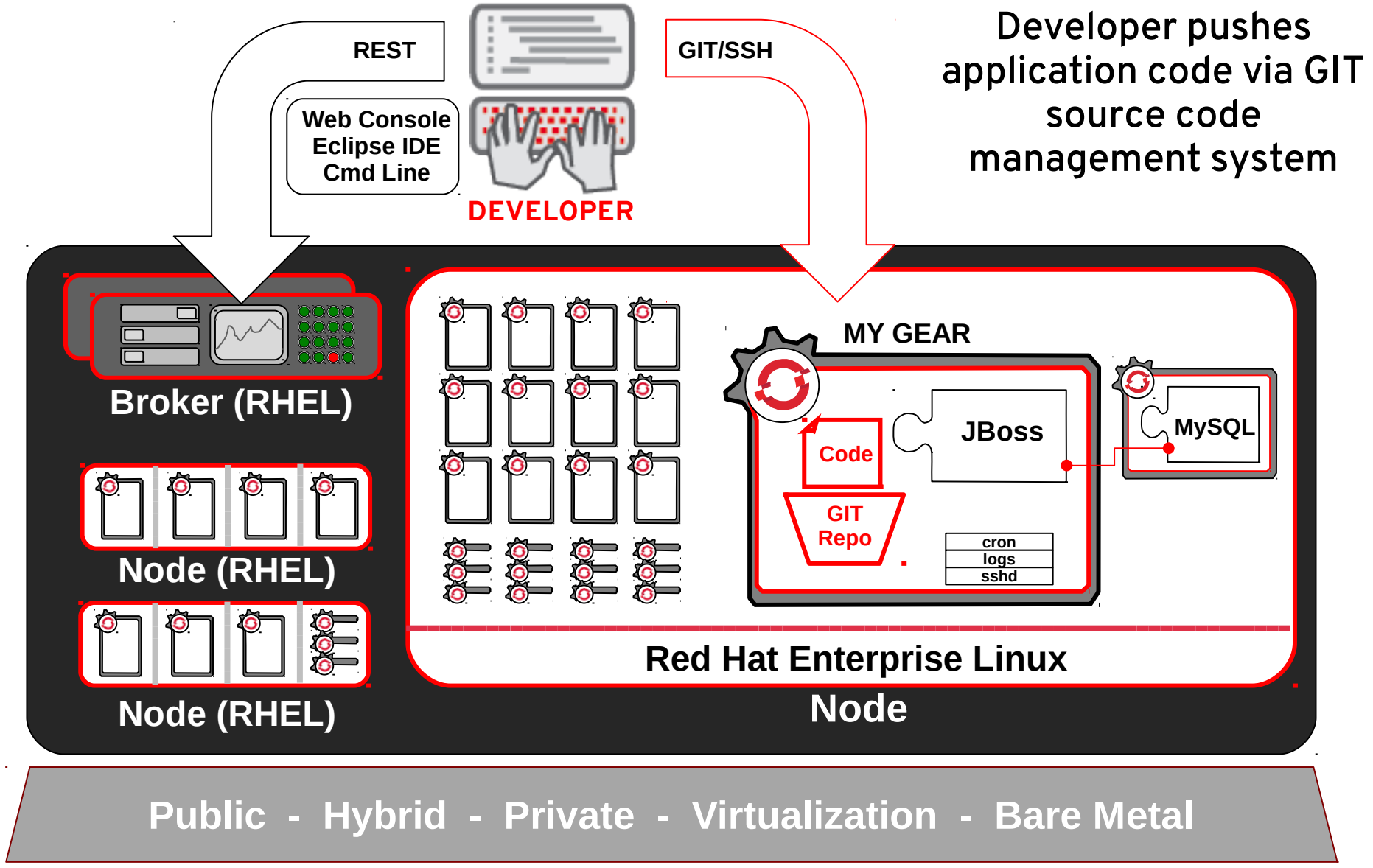
OpenShift Cartridges

Developers select a web framework cartridge
- for the desired programming language.
- handles HTTP requests and serve web pages or business APIs.
- OpenShift servers route traffic to the application

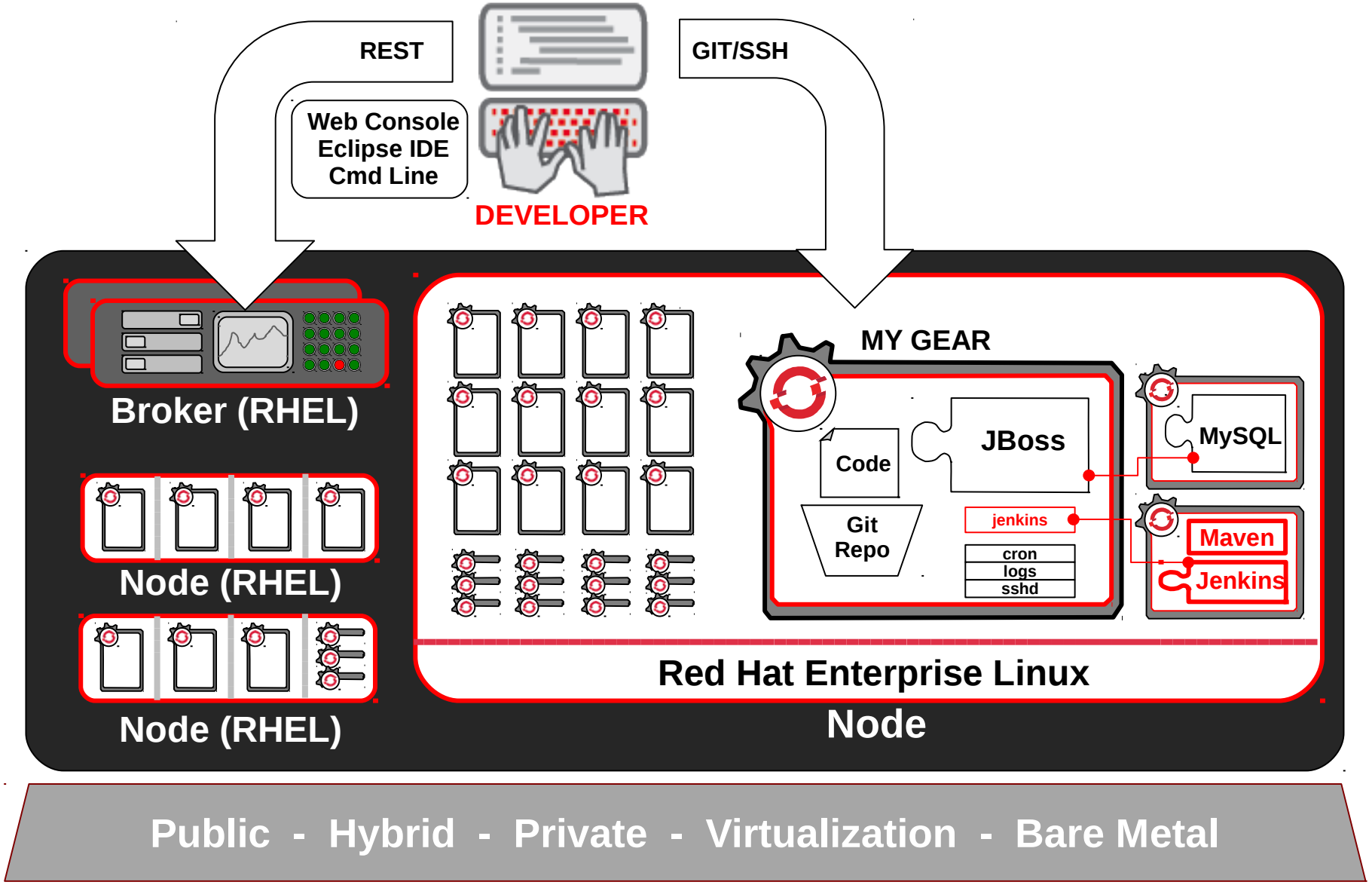
Combining cartridge
- developer can add further cartridges
  - Ex: a database or a build server
- to provide additional capabilities to the application.

Contain an implementation of the life cycle events
- **Ex: start, stop, scale...**

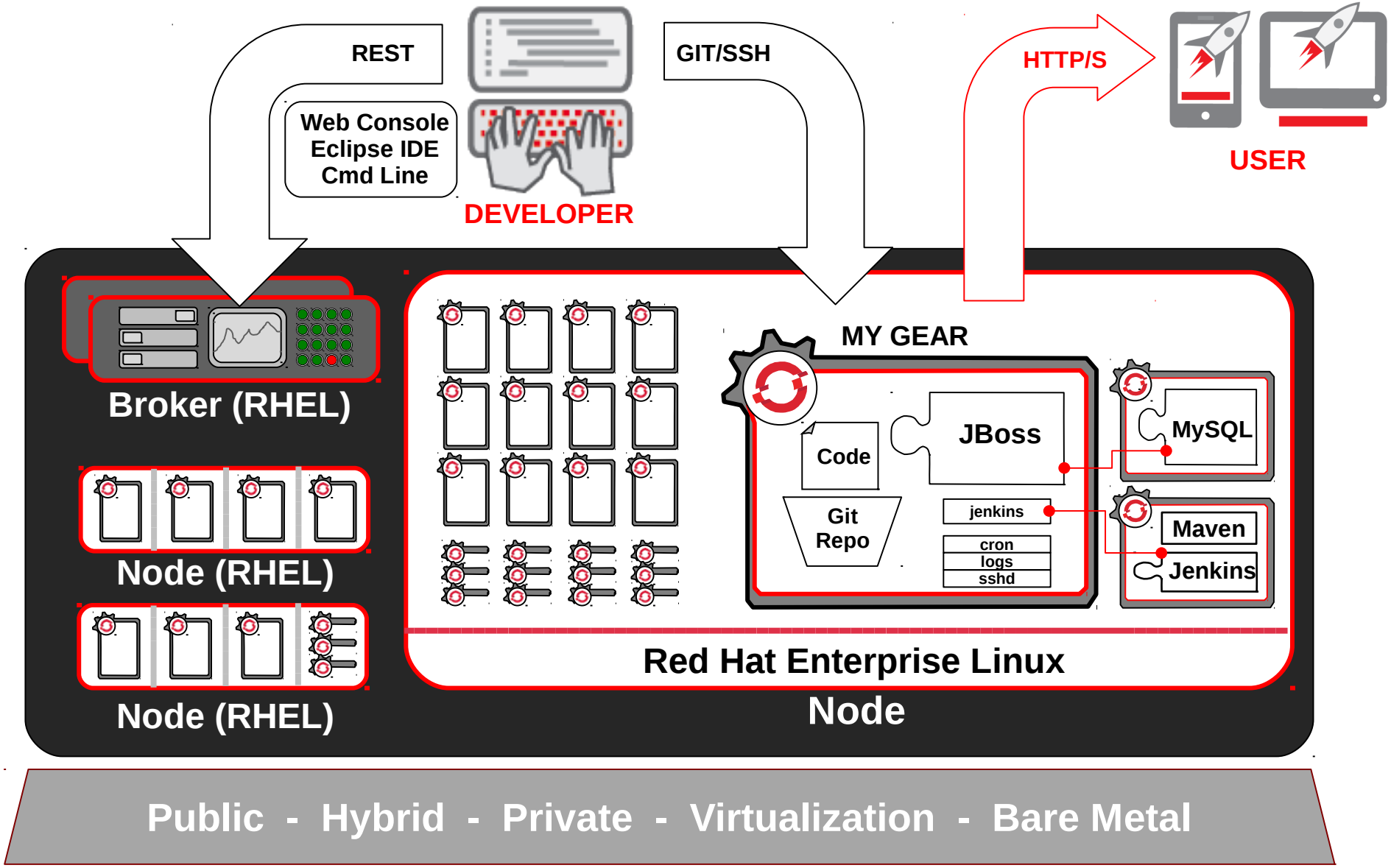
NOW, CODE AND PUSH



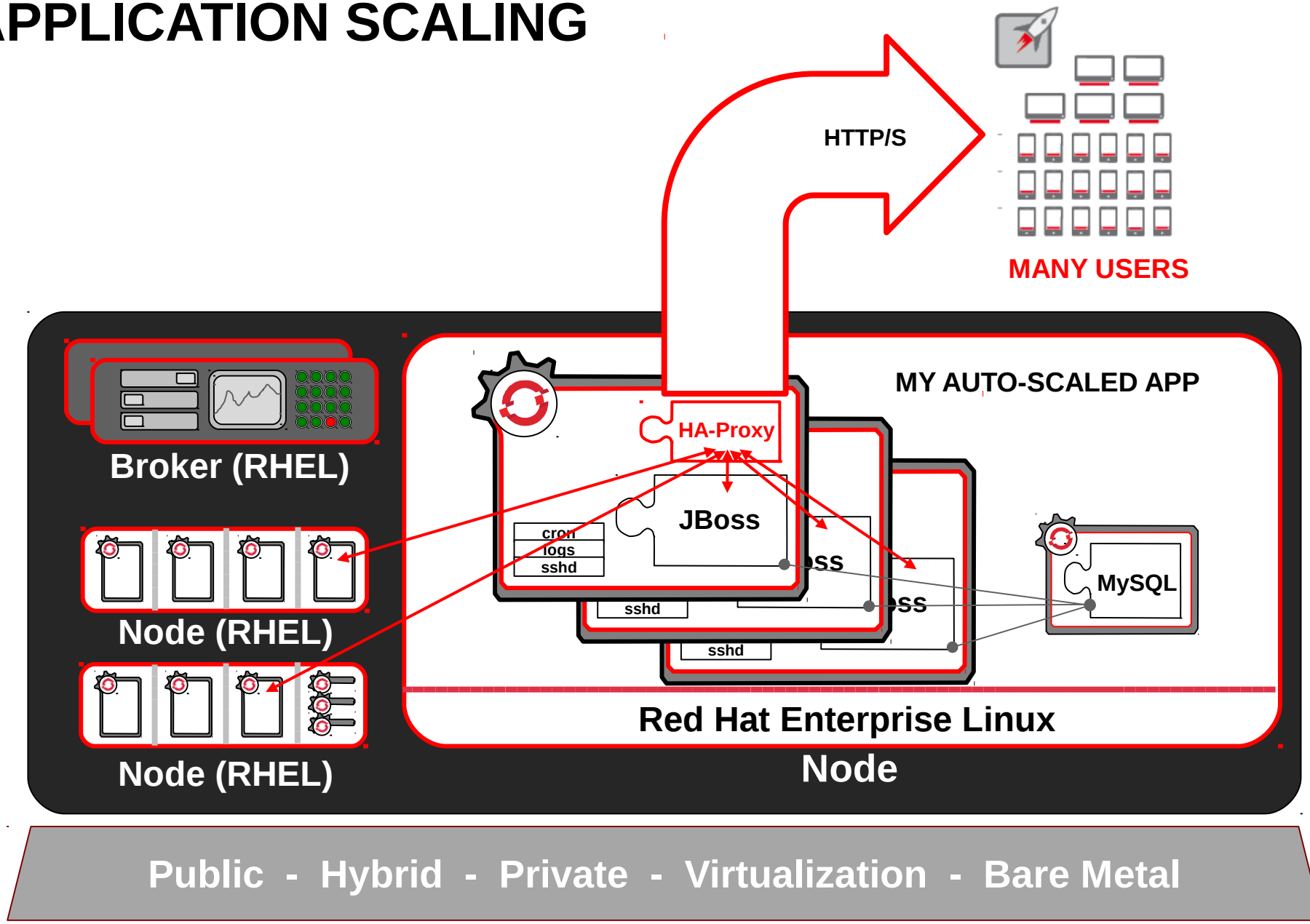
OPENSHIFT CAN AUTOMATED BUILD AND TEST WITH MAVEN AND JENKINS FOR CI



HTTP(S) SERVED FROM GEARS



OPENSHIFT AUTOMATES APPLICATION SCALING



STREAMLINING DEVELOPMENT WITH PAAS

PHYSICAL

1. Have Idea
2. Get Budget
3. Submit Hardware Request
4. Wait…
5. Get Hardware
6. Rack and Stack Hardware
7. Install Operating System
8. Install Operating System Patches
9. Create User Accounts
10. Deploy Application Server
11. Deploy Framework/Tools
12. Code
13. Test
14. Buy and Configure Prod Servers
15. Push to Prod
16. Launch
17. Order More Servers to Meet Demand
18. Wait…
19. Deploy New Servers
20. Etc.

VIRTUAL

1. Have Idea
2. Get Budget
3. Submit VM Request
4. Wait…
5. Deploy Application Server
6. Deploy Framework/Tools
7. Code
8. Test
9. Configure Prod VMs
10. Push to Prod
11. Launch
12. Request VMs to Meet Demand
13. Wait…
14. Deploy New VMs
15. Etc.

WITH PAAS

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

CRAFTWORK ASSEMBLY LINE

DEPLOYMENT PIPELINE WITH OPENSHIFT