Containerized Operating Systems

-Vineet Singh

Contents:

Core OS:

CoreOS is a barebones Linux distribution designed to make large multiple-machine deployments, using different softwares and dependencies, easier to scale and easier to manage. It is built for high availability and security. It does not come with a package manager and thus requires containers such as those provided by Docker. It uses "fleet" for cluster management and "etcd" for service discovery and keeping configuration up to date across the cluster.

CoreOS Tectonic was created with a vision of a fully automated container platform that would relieve many of the burdens of day-to-day IT operations. This vision will now help craft the next generation of Red Hat OpenShift Container Platform, providing an advanced container experience for operators and developers alike.

With automated operations coming to OpenShift, IT teams will be able to use the automated upgrades of Tectonic paired with the reliability, support, and extensive application development capabilities of Red Hat OpenShift Container Platform. This makes managing large Kubernetes deployments easier without sacrificing other enterprise needs, including platform stability or continued support for existing IT assets.

We believe this future integrated platform will help to truly change the way IT teams deliver applications by providing speed to market through consistent deployment methods and automated operations throughout the stack.

In the meantime, current Tectonic customers will continue to receive support and updates for the platform. They can also have confidence that they will be able to transition to Red Hat OpenShift Container Platform in the future with little to no disruption, as almost all Tectonic features will be retained in Red Hat OpenShift Container Platform.

Unlike most Linux distributions, CoreOS doesn’t have a package manager. Instead it takes a page from Google’s ChromeOS and automates software updates to ensure better security and reliability of machines and containers running on clusters. Both operating system updates and security patches are regularly pushed to CoreOS Container Linux machines without sysadmin intervention.

You control how often patches are pushed using CoreUpdate, with its web-based interface. This enables you to control when your machines update, and how quickly an update is rolled out across your cluster.

Specifically, CoreOS does this with the the distributed configuration service etcd. This is an open-source, distributed key value store based on YAML. Etcd provides shared configuration and service discovery for Container Linux clusters.

This service runs on each machine in a cluster. When one server goes down, say to update, it handles the leader election so that the overall Linux system and containerized applications keep running as each server is updated.

To handle cluster management, CoreOS used to use fleet. This ties together systemd and etcd into a distributed init system. While fleet is still around, CoreOS has joined etcd with Kubernetes container orchestration to form an even more powerful management tool.

CoreOS also enables you to declaratively customize other operating system specifications, such as network configuration, user accounts, and systemd units, with cloud-config.

Put it all together and you have a Linux that's constantly self-updating to the latest patches while giving you full control over its configuration from individual systems to thousand of container instances. Or, as CoreOS puts it, "You'll never have to run Chef on every machine in order to change a single config value ever again."

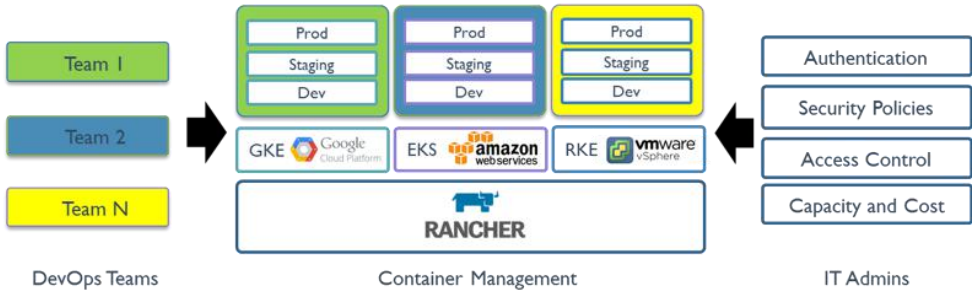
Let's say you want to expand your DevOps control even further. CoreOS helps you there, too, by making it easy to deploy Kubernetes.

So, what does all this mean? CoreOS is built from the ground-up to make it easy to deploy, manage and run containers. Yes, other Linux distributions, such as the Red Hat family with Project Atomic, also enable you to do this, but for these distributions, it's an add-on. CoreOS was designed from day one for containers.

Rancher OS:

Rancher is a container management platform built for organizations that deploy containers in production. Rancher makes it easy to run Kubernetes everywhere, meet IT requirements, and empower DevOps teams.

Rancher users have the choice of creating Kubernetes clusters with Rancher Kubernetes Engine (RKE) or cloud Kubernetes services, such as GKE, AKS, and EKS. Rancher users can also import and manage their existing Kubernetes clusters created using any Kubernetes distribution or installer.



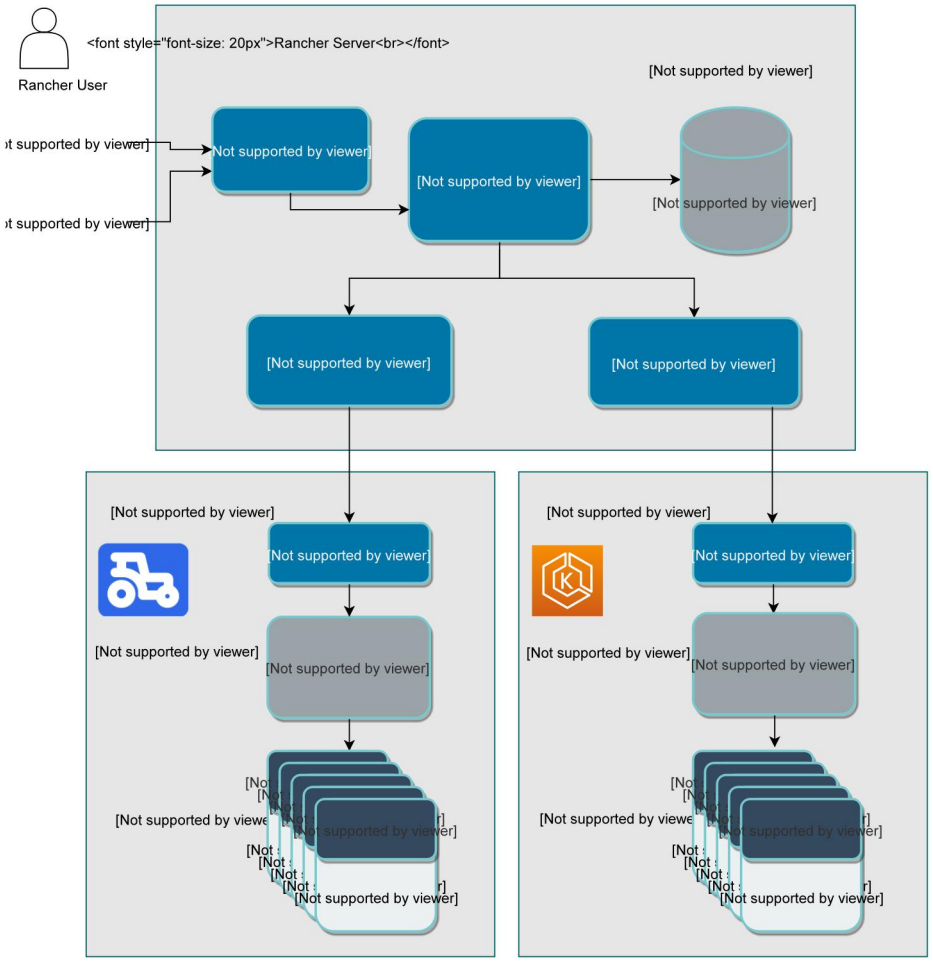
Architecture:

The majority of Rancher 2.x software runs on the Rancher Server. Rancher Server includes all the software components used to manage the entire Rancher deployment.

Rancher Server installation manages two downstream Kubernetes clusters: one created by RKE and another created by Amazon EKS (Elastic Kubernetes Service).

For the best performance and security, we recommend a dedicated Kubernetes cluster for the Rancher management server. Running user workloads on this cluster is not advised. After deploying Rancher, you can create or import clusters for running your workloads.

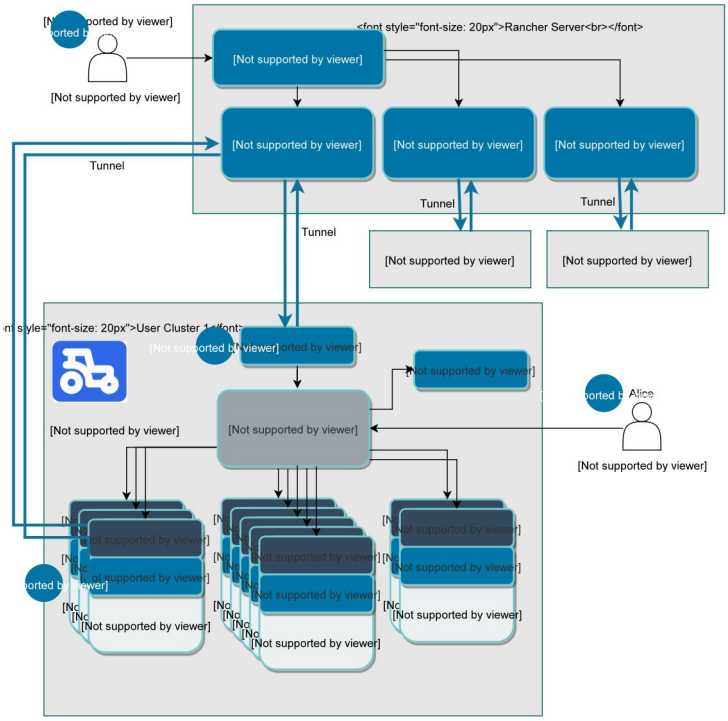
The diagram below shows how users can manipulate both Rancher-launched Kubernetes clusters and hosted Kubernetes clusters through Rancher's authentication proxy:



You can install Rancher on a single node, or on a high-availability Kubernetes cluster.
A high-availability Kubernetes installation is recommended for production. A Docker installation may be used for development and testing purposes, but there is no migration path from a single-node to a high-availability installation. Therefore, you may want to use a Kubernetes installation from the start.
The Rancher server, regardless of the installation method, should always run on nodes that are separate from the downstream user clusters that it manages. If Rancher is installed on a high-availability Kubernetes cluster, it should run on a separate cluster from the cluster(s) it manages.

Communicating with Downstream User Clusters

This section describes how Rancher provisions and manages the downstream user clusters that run your apps and services.
The below diagram shows how the cluster controllers, cluster agents, and node agents allow Rancher to control downstream clusters.
Communicating with Downstream Clusters



Rancher Components
The following descriptions correspond to the numbers in the diagram above:

1. The Authentication Proxy
In this diagram, a user named Bob wants to see all pods running on a downstream user cluster called User Cluster 1. From within Rancher, he can run a kubectl command to see the pods. Bob is authenticated through Rancher's authentication proxy.
The authentication proxy forwards all Kubernetes API calls to downstream clusters. It integrates with authentication services like local authentication, Active Directory, and GitHub. On every Kubernetes API call, the authentication proxy authenticates the caller and sets the proper Kubernetes impersonation headers before forwarding the call to Kubernetes masters.
Rancher communicates with Kubernetes clusters using a service account, which provides an identity for processes that run in a pod.
By default, Rancher generates a kubeconfig file that contains credentials for proxying through the Rancher server to connect to the Kubernetes API server on a downstream user cluster. The kubeconfig file (kube_config_rancher-cluster.yml) contains full access to the cluster.

2. Cluster Controllers and Cluster Agents

Each downstream user cluster has a cluster agent, which opens a tunnel to the corresponding cluster controller within the Rancher server.

There is one cluster controller and one cluster agent for each downstream cluster. Each cluster controller:

- Watches for resource changes in the downstream cluster
- Brings the current state of the downstream cluster to the desired state
- Configures access control policies to clusters and projects
- Provisions clusters by calling the required Docker machine drivers and Kubernetes engines, such as RKE and GKE

By default, to enable Rancher to communicate with a downstream cluster, the cluster controller connects to the cluster agent. If the cluster agent is not available, the cluster controller can connect to a node agent instead.

The cluster agent, also called cattle-cluster-agent, is a component that runs in a downstream user cluster. It performs the following tasks:

- Connects to the Kubernetes API of Rancher-launched Kubernetes clusters
- Manages workloads, pod creation and deployment within each cluster
- Applies the roles and bindings defined in each cluster's global policies
- Communicates between the cluster and Rancher server (through a tunnel to the cluster controller) about events, stats, node info, and health

3. Node Agents

If the cluster agent (also called cattle-cluster-agent) is not available, one of the node agents creates a tunnel to the cluster controller to communicate with Rancher.

The cattle-node-agent is deployed using a DaemonSet resource to make sure it runs on every node in a Rancher-launched Kubernetes cluster. It is used to interact with the nodes when performing cluster operations. Examples of cluster operations include upgrading the Kubernetes version and creating or restoring etcd snapshots.

4. Authorized Cluster Endpoint

An authorized cluster endpoint allows users to connect to the Kubernetes API server of a downstream cluster without having to route their requests through the Rancher authentication proxy.

The authorized cluster endpoint only works on Rancher-launched Kubernetes clusters. In other words, it only works in clusters where Rancher used RKE to provision the cluster. It is not available for imported clusters, or for clusters in a hosted Kubernetes provider, such as Amazon's EKS.

There are two main reasons why a user might need the authorized cluster endpoint:

- To access a downstream user cluster while Rancher is down
- To reduce latency in situations where the Rancher server and downstream cluster are separated by a long distance

The kube-api-auth microservice is deployed to provide the user authentication functionality for the authorized cluster endpoint. When you access the user cluster using kubectl, the cluster's Kubernetes API server authenticates you by using the kube-api-auth service as a webhook.

Like the authorized cluster endpoint, the kube-api-auth authentication service is also only available for Rancher-launched Kubernetes clusters.

With this endpoint enabled for the downstream cluster, Rancher generates an extra Kubernetes context in the kubeconfig file in order to connect directly to the cluster. This file has the credentials for kubectl and helm.

You will need to use a context defined in this kubeconfig file to access the cluster if Rancher goes down. Therefore, we recommend exporting the kubeconfig file so that if Rancher goes down, you can still use the credentials in the file to access your cluster. For more information, refer to the section on accessing your cluster with kubectl and the kubeconfig file.

Windows Core OS (WCOS):

Windows Core OS is a universal base from where different flavors of Windows can be created. WCOS is a shareable and modular base which means Microsoft can take it and add additional features on top of it for devices having different form factors. Windows Core OS is not an operating system in the traditional sense (like Windows 10 or 7), but a modular base which will power a host of future Windows operating systems.

Windows Core OS Features:

1. CShell (Composable Shell): CShell is a modular user-interface which can be slapped on devices depending on their form factor.
if you are using a Windows Core OS-powered laptop and if you open the Xbox Game Bar, you will instantly switch to the Xbox OS interface due to a separate CShell for gaming mode. This will make your experience much more cohesive and streamlined as if you are using an Xbox Console. The whole idea of CShell is to make the experience consistent across all Microsoft devices by making the user-interface portable.
2. Shareable Component:Microsoft is removing almost all the components to make WCOS just a barebone base. All the app layers, Libraries, and Composers are available in the form of additional components. In fact, the interface that is CShell is also available as a Component to make WCOS completely light and modular.

OS structure of "Polaris" + WCOS

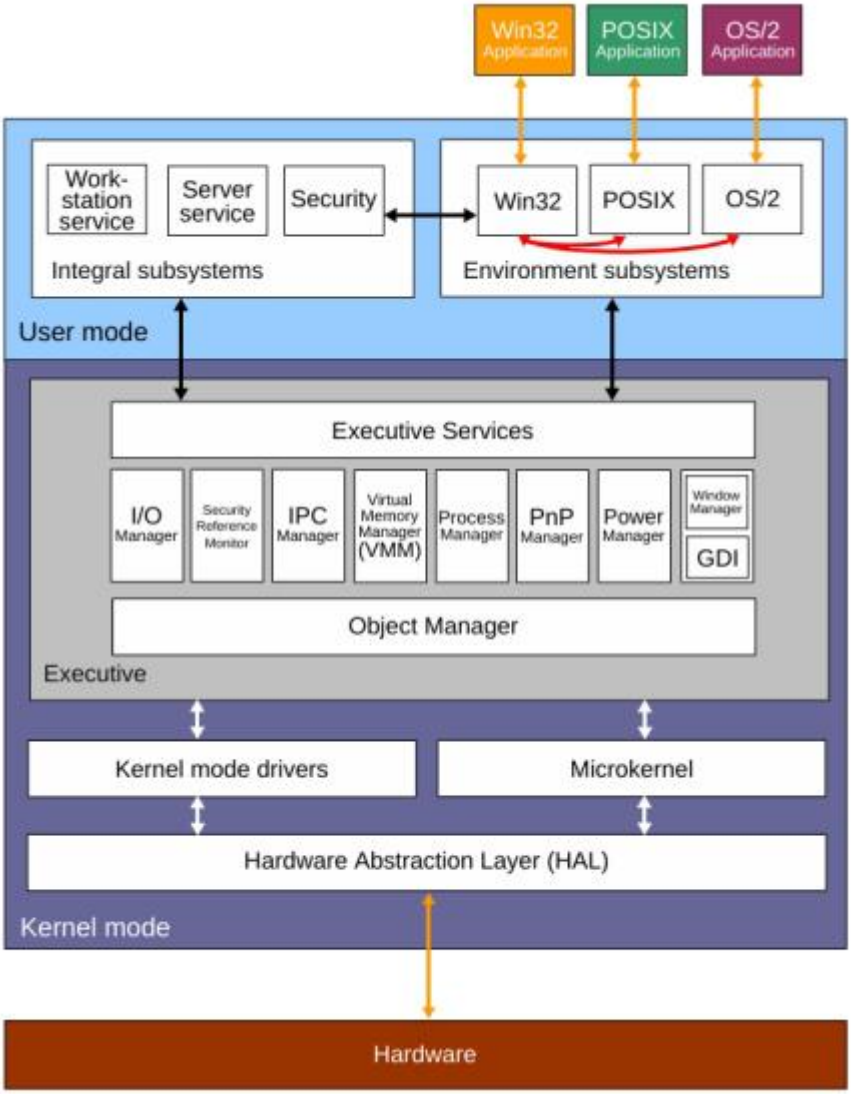
Windows Central

for instance, if they want to create an operating system for foldable phones, they can take the WCOS base and add components as required. No need to re-write and modify a huge chunk of code to make a compatible Windows operating system for different devices. This will enormously save crucial time and resources for Microsoft. Further, they can choose to add Win32 app support (as a component) on WCOS-powered laptops or leave the support on WCOS-powered tablets. Basically, share components where it makes sense

3. Faster Updates:Faster Update is a feature that many Windows users hope to see one day and it's finally coming with Windows Core OS. It's an inherent WCOS feature

so any OS built using its base will support faster update by default. This feature is straight out of Chrome OS,It will use a separate partition to install updates while you are using your device and will switch the active boot slot after a reboot. No need to wait for installation. Apart from that, Windows Core OS will use the Full Flash Update (FFU) image format to install Windows updates as opposed to ISO which will significantly reduce installation time

4. App Support:Microsoft can add an additional component for Win32 app support which will work in a container and will be completely sandboxed– just like Linux on Chrome OS. All it means for end-users is that you will be able to use your favorite Win32 apps, but the performance might take a hit since it's not running natively. But Microsoft has stated that the performance will be quite good and usable in a container.



Apart from that, WCOS will have inherent support for UWP and Web apps. As the world is increasingly moving towards web and cloud computing.

Red Hat Atomic Host:

Red Hat Enterprise Linux Atomic Host (RHEL Atomic Host) is a variant of Red Hat Enterprise Linux Server (RHEL Server) designed and optimized to run Linux Containers. On the surface both variants look quite similar.

Technical Details:RHEL Atomic Host is built around the design principles of distributed systems. When building clusters of container hosts, configuration consistency and incremental scalability are critical. These principles allow administrators utilize commodity hardware or cloud servers to quickly recover crashed nodes or add nodes as capacity to the cluster when they need to scale up. To achieve configuration consistency and incremental scalability, RHEL Atomic Host starts with some specific design constraints - a read only file system, a minimal package set, and a single command to manage host upgrades/downgrades.

1. First, the read only file system helps ensure consistency across the entire environment because unwanted file system changes are prohibited. Updates are delivered, not through RPMs, but via immutable, bootable sets called operating system trees (ostrees). Red Hat Engineering builds these ostrees from RPM content, then tests, and releases them as immutable objects. This ensures that a RHEL Atomic Host is always upgraded from a known, good starting point, to a known good destination. Having the exact same package set on all of the hosts ensures smooth and effortless upgrades, even in very large clusters - less permutation, means less problems. RHEL Atomic Host does not include yum or any other package manager because updates and rollbacks are performed transactionally with the "atomic host" command.
2. Second, RHEL Atomic Host has a lean package set - this helps ensure consistency throughout a production environment, minimizes permutations, and makes it small, quick, secure, and manageable. The packages used in RHEL Atomic Host ostrees are chosen from a subset of the exact same RPM content used in RHEL Server - this ensures the same quality of software for RHEL Atomic Host and ensures kernel and user space compatibility. The set of packages is comparable to a Minimal Install of RHEL Server, but not identical - here is an approximate comparison of packages between RHEL Server vs RHEL Atomic Host packages. All updates and rollbacks of ostrees are managed with the atomic command.
3. Finally, the atomic command provides a unified entrypoint to managing both RHEL Atomic Host as well as the container images which are cached, installed, and run locally. To upgrade RHEL Atomic Host, an administrator uses the atomic host upgrade - this command, in turn, leverages the rpm-ostree tooling to download ostree layers from a repository managed by Red Hat. It's worth noting that only the file-level delta between the running ostree and the upgrade are transferred. This is incredibly efficient from a network perspective and beneficial for minimizing bandwidth usage for disperse systems. When you upgrade, the latest ostree is made available and then deployed the next time you reboot. By default, the system keeps the newest two ostree layers locally downloaded and if there is an emergency in your cluster, you can use the atomic host rollback command to recover.

Pre installed Tools:

4. RHEL Atomic Host comes pre-installed with all the tools for running containers - docker, runc, kubernetes client, etcd, flannel, atomic. Orchestrating containers via Kubernetes is supported in a single master/node deployment on a single

system. Larger, multi-system environments use OpenShift Container Platform to provide multi-host, container orchestration. Additional details about supported container orchestration tools can be found at How are container orchestration tools supported with Red Hat Enterprise Linux?.

5. RHEL Atomic Host has comes pre-installed with container performance tuning profiles, Ceph & Gluster storage clients, SSSD, and iSCSI tools.

Operating System Content

6. The operating system content is delivered as an immutable ostree. Any additional software not found in the ostree should be run from inside of a container. This may require the use of the --privileged docker option or a Super-Privileged Container such as the RHEL Atomic Tools container for software that cannot be run from a container.

7. The yum command is not present in RHEL Atomic Host and cannot be used to install packages. However, it is still possible to use the rpm command to query packages installed in the immutable ostree. Note that Yum is available inside RHEL-based container images.

8. As a fall back, it is possible to install RPMs on the system using the atomic install command. This is highly useful for troubleshooting a system that is misbehaving.

9. RHEL Atomic Host supports atomic upgrades and rollbacks of the OS. You can power off the system during an upgrade/rollback operation and it will still be functional during the next boot. RHEL Server is not as fault-tolerant during upgrades and does not have robust rollback support. This is useful in large distributed systems.

10. On RHEL Atomic Host, there are two new directories in root (/): the /sysroot/ directory, and the /ostree/ directory.

11. The atomic host deploy command is available on Atomic Host which lets you choose a specific version of an ostree, bringing more flexibility than upgrade and rollback. For example, you can the following command to deploy a particular version of RHEL Atomic Host:
    Raw
    # atomic host deploy 7.2.7
12. No man pages - To save storage space, manual pages are not shipped with the Atomic Host image. However, the RHEL Atomic Tools container has the man pages for the packages that make up the ostree and you can access them by running the container:
    Raw
    # atomic run rhel7/rhel-tools man rpm-ostree

System Management

13. Cockpit is a powerful, modern web UI for RHEL and RHEL Atomic Host. The web frontend is delivered as a container for Atomic Host and provides an interface that makes administering containers a breeze for admins who come from a virtualization background.

14. From the boot prompt, Atomic Host has the ability to go straight into the Cockpit UI via Developer Mode. This provides developers with configuration free access to get started with containers. See Chapter 2 of the Red Hat Enterprise Linux Atomic Host 7 Installation and Configuration Guide for additional information about Developer Mode.

15. There are only two writable directories for local system configuration: /etc/ and /var/. All other directories on the system are read-only. User and host specific data that is intended to persist across updates should be stored only in the /var/ directory. For example, the /home directory is symlinked to /var/home and therefore is writable as usual. Configuration files in the /etc/ directory can still be modified as usual.'

16. The storage configuration of RHEL Atomic Host uses LVM by default. During installation, two Logical Volumes (LV) are created. The root LV is used for the operating system content. The "docker-pool" LV is thinly-provisioned and is configured to grow automatically. The "docker-pool" LV is used as the storage backend for Docker. The Docker storage configuration is managed via the atomic CLI and/or Cockpit.

17. RHEL Atomic Host provides a choice between docker and docker-latest, but Red Hat does not support running both docker and docker-latest on the same machine at the same time.

18. There are two new tuned profiles optimized for Atomic, atomic-host for physical machines and atomic-guest for virtual machines.

Common Strengths

19. Security - SELinux is enabled by default on both RHEL Server and RHEL Atomic Host. SELinux provides strong safeguards in multi-tenant environments. The iptables services are available as a firewall, iptables is turned off by default. The atomic scan command is available on both systems (on RHEL Atomic Host it is pre-installed) to check if your containers comply with the security policies, and if they have vulnerabilities.

20. Support - RHEL Atomic Host is included with RHEL Server subscriptions at no extra cost. End users can gain access to support through phone or the Red Hat portal with the same level of support as RHEL Server.

21. Same RPMs - RHEL Atomic Host is built using a subset of the exact same RPMs that are used in RHEL Server, however their content is delivered in the form of an immutable ostree. This allows administrators to leverage their RHEL

knowledge but still gain the manageability of an immutable system in a large distributed system.

Photon OS:

Photon OS was developed with an environment that provided consistency from development through production, to smooth integration and deployment and speed time to market.
It is a lightweight Linux operating system for cloud-native apps. Photon is optimized for vSphere and vCloud Air, providing an easy way for our customers to extend their current platform with VMware and run modern, distributed applications using containers.

Photon provides the following benefits:

- Support for the most popular Linux container formats including Docker, rkt, and Garden from Pivotal
- Minimal footprint (approximately 300MB), to provide an efficient environment for running containers
- Seamless migration of container workloads from development to production
- All the security, management, and orchestration benefits already provided with vSphere offering system administrators with operational simplicity

Features:

The two distinguishing features of Photon OS are as follows:

- It manages services with systemd.
  By using systemd, Photon OS adopts a contemporary Linux standard to manage system services. Photon OS bootstraps the user space and concurrently starts services with systemd. The systemctl utility controls services on Photon OS. For example, instead of running the /etc/init.d/ssh script to stop and start the OpenSSH server on a init.d-based Linux system, you run the following systemctl commands on Photon OS:
  - systemctl stop sshd
  - systemctl start sshd
- It manages packages with an open source, yum-compatible package manager called tdnf for Tiny DNF.
  Tdnf keeps the operating system as small as possible while preserving yum's robust package-management capabilities. On Photon OS, tdnf is the default package manager for installing new packages. It is a C implementation of the DNF package manager.

Installer Updates

- Deployment using RPM OStree.
- Network configuration support using the installer.
- LVM support for root partition.
- Trusted Platform Module Support (TPM).
- Ability to run installer from multiple media such as USB, CDROM, kickstart etc. on to a wider range of storage devices.

Package and Binary Maintenance

- Cloud-ready images for rapid deployment on Microsoft Azure (new), Google Compute Engine (GCE), Amazon Elastic Compute Cloud (EC2), and VMware products (vSphere, Fusion, and Workstation)
- Critical updates to the following base OS packages:

- Linux kernel 4.19
- Glibc 2.28
- systemd 239
- Python3 3.7
- Openjdk : 1.8.0.232, 1.11.0.28 and 1.10.0.23
- Openssl : 1.0.2t and 1.1.1d
- Cloud-init: 19.1
- Up-to-date versions for most packages available in the repository.
- Ability to support multiple versions of the same package (For example, go-1.9, go-1.10, go-1.11 and go-1.13).
- Support for new packages including Ostree, tpm2-tss, tpm2-tools, tpm2-abrmd and so on.

Summary:

Core OS

1. Doesn't comes with package manger and require containers ,provided by docker.
2. It uses *fleet* for cluster management and *etcd* for service discovery and keeping configuration up to date across the cluster.
3. It uses *Red Hat Openshift* container platform for application deployment.
4. The user have control over the updates that are being pushed in system using *coreupdates.*
5. *Etcd* is the distributed configuration service written with YAML. This service runs on every machine on a cluster, it ensures the whole production is up even when one machine is updating.
6. *Fleet* it ties *systemd* and *etcd* to handle cluster management.

Rancher OS

7. Users can create *kubernetes cluster(RKE)* or or cloud Kubernetes services, such as GKE, AKS, and EKS. Rancher users can also import and manage their existing Kubernetes clusters created using any Kubernetes distribution or installer.
8. Rancher Server installation manages two downstream Kubernetes clusters: one created by *RKE* and another created by Amazon *EKS (Elastic Kubernetes Service).*
9. *Authentication Proxy* forwards all kubernetes API calls to downstream clusters and is integrated with local authentication,AD,github
10. There is *one cluster controller* and one *cluster agent* for each downstream cluster.Each downstream user cluster has a cluster agent, which opens a tunnel to the corresponding cluster controller within the rancher server.
11. If the cluster agent (also called cattle-cluster-agent) is not available, one of the *node agents* creates a tunnel to the cluster controller to communicate with Rancher.
12. An authorized cluster endpoint allows users to connect to the Kubernetes API server of a downstream cluster without having to route their requests through the Rancher authentication proxy.
    it only works in clusters where Rancher used RKE to provision the cluster. It is not available for imported clusters, or for clusters in a hosted Kubernetes provider, such as Amazon's EKS.

Windows Core OS

13. Windows Core OS is a universal base from where different flavors of Windows can be created.
14. *CShell* is a modular user-interface which can be slapped on devices depending on their form factor.
15. Microsoft is removing all legacy component and replacing with *share able component*, like app layers, libraries and additional components.
16. *Faster Update* and *App Support* is another feature, which will provide faster and quicker updates and app deployment in form of containers.

Red Hat Atomic Host

17. It is optimized to run linux containers.
18. It allows administrators to quickly recover crashed nodes or add nodes as capacity to the cluster when they need to scale up.
19. RHEL Atomic Host comes pre-installed with all the tools for running containers - docker, runc, kubernetes client, etcd, flannel, atomic.
20. RHEL Atomic Host supports atomic upgrades and rollbacks of the OS. You can power off the system during an upgrade/rollback operation and it will still be functional during the next boot.
21. Other features include *System Management* and *Common Strengths* like selinux,RPM supports.

Photon OS

22. It is a lightweight Linux operating system for cloud-native apps.it is optimized for vSphere and vCloud Air, providing an easy way for our customers to extend their current platform with VMware and run modern, distributed applications using containers.
23. It manages service with *systemd,* it manages packages with yum-compatible package manager called tdnf.
24. Cloud-ready images for rapid deployment on Microsoft Azure (new), Google Compute Engine (GCE), Amazon Elastic Compute Cloud (EC2), and VMware products (vSphere, Fusion, and Workstation).
25. Critical updates to the following base OS packages.