January 2023

SYSTEM AND METHOD FOR BUILDING CONTAINER CLUSTER

PANNEER PERUMAL
Visa

BALASAHEB RAOSAHEB DENGALE
Visa

CHARAN RAMIREDDY
Visa

DILSHAD T
Visa

MANJUNATH Y
Visa

See next page for additional authors

Inventor(s)
PANNEER PERUMAL, BALASAHEB RAOSAHEB DENGALE, CHARAN RAMIREDDY, DILSHAD T, MANJUNATH Y, and ADITYA SAXENA

TITLE: “SYSTEM AND METHOD FOR BUILDING CONTAINER CLUSTER”

VISA

PANNEER PERUMAL

BALASAHEB RAOSAHEB DENGALE

CHARAN RAMIREDDY

DILSHAD T

MANJUNATH Y

ADITYA SAXENA

15

TECHNICAL FIELD

[001] The present subject matter relates to a field of cloud computing, more particularly, but not exclusively to a system and method for building container clusters.

BACKGROUND

[002] Recently, there is a rise in leveraging cluster-based architectures to deploy and manage applications in cloud. A computer cluster is a group of two or more computers, or nodes, that run in parallel to achieve a common goal. This allows workloads consisting of a high number of individuals, parallelizable tasks to be distributed among the nodes in the cluster. As a result, these tasks can leverage the combined memory and processing power of each computer to increase overall performance.

[003] The clusters are designed to minimize latency and prevent bottlenecks in node-to-node communication. Thus, there are multiple clusters built across datacenters and network zones to minimize latency and manage applications in the cloud. However, building of new clusters for adding capacity, or upgrading old clusters involve complexities due to multiple vendors, products, internal integration with storage, network, cybersecurity, cloud view, and the like. Further, in existing systems, the clusters are built manually and is time-consuming. As such, there is no automated method using which new clusters may be added or managed for deploying the applications.

[004] The information disclosed in this background of the disclosure section is only for enhancement of understanding of the general background of the invention and should not be taken as an acknowledgement or any form of suggestion that this information forms the prior art already known to a person skilled in the art.

BRIEF DESCRIPTION OF THE DRAWINGS

[005] The accompanying drawings, which are incorporated in and constitute a part of this disclosure, illustrate exemplary embodiments and, together with the description, serve to explain the disclosed principles. In the figures, the left-most digit(s) of a reference number identifies the figure in which the reference number first appears. The same numbers are used

throughout the figures to reference like features and components. Some embodiments of device or system and/or methods in accordance with embodiments of the present subject matter are now described, by way of example only, and with reference to the accompanying figures, in which:

[006] Figure 1 illustrates an exemplary environment for building container clusters, in accordance with some embodiments of the present disclosure;

[007] Figure 2 illustrates a sequence diagram for building container clusters, in accordance with some embodiments of the present disclosure; and

[008] Figure 3 illustrates a block diagram of an exemplary computer system for implementing embodiments consistent with the present disclosure.

[009] The figures depict embodiments of the disclosure for purposes of illustration only. One skilled in the art will readily recognize from the following description that alternative embodiments of the structures and methods illustrated herein may be employed without departing from the principles of the disclosure described herein.

DESCRIPTION OF THE DISCLOSURE

[010] In the present document, the word "exemplary" is used herein to mean "serving as an example, instance, or illustration." Any embodiment or implementation of the present subject matter described herein as "exemplary" is not necessarily to be construed as preferred or advantageous over other embodiments.

[011] While the disclosure is susceptible to various modifications and alternative forms, specific embodiment thereof has been shown by way of example in the drawings and will be described in detail below. It should be understood, however that it is not intended to limit the disclosure to the particular forms disclosed, but on the contrary, the disclosure is to cover all modifications, equivalents, and alternative falling within the spirit and the scope of the disclosure.

[012] The terms “comprises”, “comprising”, or any other variations thereof, are intended to cover a non-exclusive inclusion, such that a setup, device, or method that comprises a list of

components or steps does not include only those components or steps but may include other components or steps not expressly listed or inherent to such setup or device or method. In other words, one or more elements in a device or system or apparatus proceeded by “comprises… a” does not, without more constraints, preclude the existence of other elements or additional elements in the device, system, or apparatus.

[013] The terms "an embodiment", "embodiment", "embodiments", "the embodiment", "the embodiments", "one or more embodiments", "some embodiments", and "one embodiment" mean "one or more (but not all) embodiments of the invention(s)" unless expressly specified otherwise.

[014] The terms "including", "comprising", “having” and variations thereof mean "including but not limited to", unless expressly specified otherwise.

[015] The present disclosure discloses a method and system for building container clusters. Generally, the container clusters are built manually and building of new container clusters for adding capacity or upgrading old container clusters involves complexity. The complexity is due to multiple vendors, products, network, cybersecurity and the like. Thus, manually building container clusters is time-consuming. To overcome the above problem, the present disclosure automates the process of building container clusters. The present disclosure automates the process of hardening base operating system/kernel. Further, present disclosure installs vendor platforms, configure network and storage as per user requirement. The present disclosure manages distribution of traffic/client connects by using load balancer without any manual intervention. Thereafter, the present disclosure performs end-to-end validations of the container cluster. Thus, the present disclosure reduces time consumption for building container cluster by automating the process.

[016] Figure 1 illustrates an exemplary environment 100 for building container clusters for managing application. The environment 100 includes a container management system 101, and a cluster node 102_1, 102_2, …., 102_n (hereafter referred as plurality of cluster nodes 102). The container management system 101 may be implemented within a datacentre. The datacentre is a facility that provides shared access to applications and data using a complex network, compute, and storage infrastructure. In an embodiment, the container management system 101 may be implemented through self-service portal such as – CloudView Scalable Run™ /

IMAGE (API gateway for infrastructure), and the like. These implementations may obtain required information from source control and users ansible tower® to trigger the building on the plurality of cluster nodes 102. In an embodiment, the ansible tower® is a web-based solution that is utilised to provision underlying infrastructure of environment, virtualized hosts, hypervisors, network devices, and the like. In an embodiment, the ansible tower is utilised for automating the task of creating the container. This includes setting up base nodes with required packages like Docker®, Simple input/output (Sio) package, selinux and so on. In an embodiment, automation may also include configuring base cluster with minimal nodes and adding new nodes as worker nodes. The automation may apply standard configurations such as, authentication with Light Weight Directory Access Protocol (LDAP), storage integration for block / file system storage with Kubernetes®, network integration with calico™/cilium™ and so on. Further, the container management system 101 may validate end to end with sample application for deploying cluster/node to application. Further, each of the plurality of cluster nodes 102 comprise pods and containers. A pod may refer to a group of one or more containers, with shared storage and network resources, and a specification related to running the containers. The plurality of cluster nodes 102 may also be referred as worker machines that run containerized applications. The containerized applications are applications that run in isolated runtime environment called the containers. The containers encapsulate an application as a single executable package of software that bundles application code together with all of the related configuration files, libraries, and dependencies required for it to operate. The plurality of cluster nodes 102 host the pods. The pods are designed to support multiple cooperating processes (containers) on a cluster. The container management system 101 manages the containers that resides on the plurality of cluster nodes 102. In an embodiment, the container management system 101 may build and manage new containers which may be added based on user requirement. Further, the container management system 101 may include one or more processor 103, I/O interface 104, and a memory 105. In some embodiments, the memory 105 may be communicatively coupled to the one or more processors 103. The memory 105 stores instructions, executable by the one or more processors 103, which, on execution, may cause the container management system 101 to build and manage container clusters, as disclosed in the present disclosure. In an embodiment, the memory 105 may include one or more modules 106 and data 107. The one or more modules 106 may be configured to perform the steps of the present disclosure using the data 107, to provide an automated process for building and

managing container clusters. In an embodiment, each of the one or more modules 106 may be a hardware unit which may be present outside the memory 105 and coupled with the container management system 101. The container management system 101 may be implemented in a variety of computing systems, such as, a laptop computer, a desktop computer, a Personal Computer (PC), a notebook, a smartphone, a tablet, e-book readers, a server, a network server, a cloud-based server, and the like. In an embodiment, the container management system 101 may be a dedicated server or may be a cloud-based server.

[017] Initially, based on the user requirement, a new container cluster may be added to the plurality of cluster nodes 102. The user requirement may include, but is not limited to, adding capacity, deploying new application, products, and the like. In an embodiment, the container management system 101 may automate hardening of the base operation system/kernel. The hardening of the kernel is a process of implementing security measures and patching for operating system to reduce security risk by eliminating potential attack vectors. The container management system 101 may install vendor platforms from stash and maintain the installed vendor platforms as infrastructure as a code. In an embodiment, configuration files related to the plurality of cluster nodes 102 may be stored in source control system such as, the stash for version control. In an embodiment, the stash may be a source of truth (i.e., a single point of reference) and may be utilised during installation of vendor platforms. Further, the container management system 101 may configure network and storage for adding the new container based on the user requirement. Then, the container management system 101 may obtain and configure load balancer Virtual Internet Protocol (VIP) without any manual intervention. In an embodiment, initially before building container clusters, shared wildcard load balancers are created by load balancer team. Thereafter, information related to the plurality of cluster nodes 102 may be stored in the source control system like stash. The container management system 101 may be configured to fetch the information during automation process of building cluster containers to configure the plurality of cluster nodes 102. The load balance VIP is utilised to distribute client connections to backend servers. In an embodiment, the load balancer VIP of the container management system 101 may track availability of pods of the plurality of cluster nodes 102 to sort/assign a request for a specific service. Further, the container management system 101 performs an end-to-end validation by deploying test application and validating the test application. In an embodiment, extensive automation may deploy sample application on each node in the plurality of cluster nodes 102 and validate if it’s working end-to-end. In an

embodiment, the container management system 101 may also perform steps to validate storage / network integrations, and the like.

[018] Figure 2 illustrates a sequence diagram for building container clusters, in accordance with some embodiments of the present disclosure. The container clusters are shared resources where one or multiple applications may be deployed which may span across multiple network zones. In an embodiment, the container cluster is created with multiple physical servers along with storage and network resources. The configurations for the clusters may be stored in the source control system such as, stash and referred by automations. The automations perform a set of activities based on type of node in sequential order with multiple nodes in parallel. Some of the activities may include storage, state of the build with facts, installing base OS packages like SIO, systats, selinux, Docker® etc. Cluster may first build the master nodes to create base minimal clusters and continue to add new workers to this cluster. This may perform cluster specific actions like LDAP configurations for users authentications, calico configurations for network, Pod Security Policies (PSP) install for security hardening, and the like. In Figure 2, initially, the container management system 101 may receive an image as input. The image may be referred as a container image. The container image is a static file with executable codes that may be used to create a container on a computing system. In an embodiment, the container image may be a set of instructions or a template to build/create the container. The container image is then provided to an ansible tower job. In an embodiment, the ansible tower® is a web-based solution that is utilised to provision underlying infrastructure of environment, virtualized hosts, hypervisors, network devices, and the like. In an embodiment, the ansible tower is utilised for automating the task of creating the container. Further, the container management system 101 may perform clone clustering configuration. That is, the container management system 101 may copy configuration information of original cluster to use as a basis for creating a new cluster to run the container. The container management system 101 may generate or configure memory usage for the new cluster. Further, the container management system 101 may initiate the process of building the container cluster. Firstly, the container management system 101 may download and install a Docker Enterprise Edition (EE). The Docker EE is designed for enterprise development for building, shipping. and running business-critical applications. After installing the docker EE, the container management system 101 may install Simple input/output (Sio) package. The Sio package is utilised for reading and writing binary data in Sio structures called record and block. The container management system 101 may then

copy relevant Docker trusted Registry (DTR) certificates. In an embodiment, the DTR may be installed to securely store and manage images used in the applications. Further, the content management system 101 may check the type of operating system. In an embodiment, if the operating system is a Red Hat® Linux®, then selinux settings is applied. While if the operating system is a Debian®, selinux setting is not applied. The selinux may be a kernel security module to manage access policies related to resources like file system. The container management system 101 may install and enable systat package. The systat package is a statistics and statistical graphics software package for monitoring system resources, their performance and usage activities. The container management system 101 may apply patches for Grand Unified Bootloader (GRUB) configuration. The GRUB is a tool for booting and loading operating system kernels. The container management system 101 may adjust vm.max-map-count and download latest kubectl®. In an embodiment, the kubectl® is a Kubernetes command-line tool that allows to run commands, deploy applications, inspect, and manage cluster resources, and the like. The container management system 101 deploys Universal Control Plane (UCP) in primary master node. Further, the container management system 101 applies UCP Tom's Obvious Minimal Language (TOML) configuration file and adjusts Lightweight Directory Access Protocol (LDAP) configuration on the UCP. The LDAP provides communication language that application use to communicate with directory service servers. The TOML may be designed to map unambiguously to a hash table. Further, the TOML may be easy to parse into data structures in different languages. In an embodiment, the UCP TOML may be UCP configuration file for the plurality of cluster nodes 102. The container management system 101 may add manager nodes and worker nodes. Further, the container management system 101 may copy and enable a job for daily UCP-backup. The container management system 101 downloads client-bundle in UCP-primary and apply Pod Security Policies (PSPs). The PSPs is a cluster-level resource that controls the actions that a pod can perform and access. Further, the container management system 101 may copy calico setup and configuration files and apply labels to each of the plurality of cluster nodes 102. Calico is an open-source networking and network security solution for containers, virtual machines, and native host-based workloads. Further, the container management system 101 may check type of Container Network Interface (CNI). If the CNI type is calico-vxlan, the container management system 101 may install calico vxlan type CNI and apply calico vxlan configuration. In an embodiment, if the CNI type is BGP-peer, the container management

system 101 may install CNI of type BGP-peer and apply BGP-peer configuration for each node. Further, the container management system 101 may restart docker service. Thereafter, the container management system 101 may perform basic validation and update status for the image.

Computing System

[019] Figure 3 illustrates a block diagram of an exemplary computer system 300 for implementing embodiments consistent with the present disclosure. In an embodiment, the computer system 300 is used to implement the container management system 101 for building container cluster. The computer system 300 may include a central processing unit ("CPU" or "processor") 302. The processor 302 may include at least one data processor for executing processes in Virtual Storage Area Network. The processor 302 may include specialized processing units such as, integrated system (bus) controllers, memory management control units, floating point units, graphics processing units, digital signal processing units, etc.

[020] The processor 302 may be disposed in communication with one or more input/output (I/O) devices 309 and 310 via I/O interface 301. The I/O interface 301 may employ communication protocols/methods such as, without limitation, audio, analog, digital, monaural, RCA, stereo, IEEE-1394, serial bus, universal serial bus (USB), infrared, PS/2, BNC, coaxial, component, composite, digital visual interface (DVI), high-definition multimedia interface (HDMI), radio frequency (RF) antennas, S-Video, VGA, IEEE 802.n /b/g/n/x, Bluetooth, cellular (e.g., code-division multiple access (CDMA), high-speed packet access (HSPA+), global system for mobile communications (GSM), long-term evolution (LTE), WiMax, or the like), etc.

[021] Using the I/O interface 301, the computer system 300 may communicate with one or more I/O devices 309 and 310. For example, the input devices 309 may be an antenna, keyboard, mouse, joystick, (infrared) remote control, camera, card reader, fax machine, dongle, biometric reader, microphone, touch screen, touchpad, trackball, stylus, scanner, storage device, transceiver, video device/source, etc. The output devices 310 may be a printer, fax machine, video display (e.g., cathode ray tube (CRT), liquid crystal display (LCD), light-emitting diode (LED), plasma, Plasma Display Panel (PDP), Organic light-emitting diode display (OLED) or the like), audio speaker, etc.

[022] In some embodiments, the computer system 300 may consist of the container management system 101. The processor 302 may be disposed in communication with a communication network 311 via a network interface 03. The network interface 303 may communicate with the communication network 311. The network interface 303 may employ connection protocols including, without limitation, direct connect, Ethernet (e.g., twisted pair 10/100/1000 Base T), transmission control protocol/internet protocol (TCP/IP), token ring, IEEE 802.11a/b/g/n/x, etc. The communication network 311 may include, without limitation, a direct interconnection, local area network (LAN), wide area network (WAN), wireless network (e.g., using Wireless Application Protocol), the Internet, etc. Using the network interface 303 and the communication network 311, the computer system 300 may communicate with a plurality of cluster nodes 312 to provide automation process for building container cluster. The network interface 303 may employ connection protocols include, but not limited to, direct connect, Ethernet (e.g., twisted pair 10/100/1000 Base T), transmission control protocol/internet protocol (TCP/IP), token ring, IEEE 802.11a/b/g/n/x, etc.

[023] The communication network 311 includes, but is not limited to, a direct interconnection, an e-commerce network, a peer to peer (P2P) network, local area network (LAN), wide area network (WAN), wireless network (e.g., using Wireless Application Protocol), the Internet, Wi-Fi, and such. The first network and the second network may either be a dedicated network or a shared network, which represents an association of the different types of networks that use a variety of protocols, for example, Hypertext Transfer Protocol (HTTP), Transmission Control Protocol/Internet Protocol (TCP/IP), Wireless Application Protocol (WAP), etc., to communicate with each other. Further, the first network and the second network may include a variety of network devices, including routers, bridges, servers, computing devices, storage devices, etc.

[024] In some embodiments, the processor 302 may be disposed in communication with a memory 305 (e.g., RAM, ROM, etc. not shown in **Figure 3**) via a storage interface 304. The storage interface 304 may connect to memory 305 including, without limitation, memory drives, removable disc drives, etc., employing connection protocols such as, serial advanced technology attachment (SATA), Integrated Drive Electronics (IDE), IEEE-1394, Universal Serial Bus (USB), fibre channel, Small Computer Systems Interface (SCSI), etc. The memory

drives may further include a drum, magnetic disc drive, magneto-optical drive, optical drive, Redundant Array of Independent Discs (RAID), solid-state memory devices, solid-state drives, etc.

[025] The memory 305 may store a collection of program or database components, including, without limitation, user interface 306, an operating system 307, web browser 308 etc. In some embodiments, computer system 300 may store user/application data, such as, the data, variables, records, etc., as described in this disclosure. Such databases may be implemented as fault-tolerant, relational, scalable, secure databases such as Oracle ® or Sybase®.

[026] The operating system 307 may facilitate resource management and operation of the computer system 300. Examples of operating systems include, without limitation, APPLE MACINTOSH® OS X, UNIX®, UNIX-like system distributions (E.G., BERKELEY SOFTWARE DISTRIBUTION™ (BSD), FREEBSD™, NETBSD™, OPENBSD™, etc.), LINUX DISTRIBUTIONS™ (E.G., RED HAT™, UBUNTU™, KUBUNTU™, etc.), IBM™ OS/2, MICROSOFT™ WINDOWS™ (XP™, VISTA™/7/8, 10 etc.), APPLE® IOS™, GOOGLE® ANDROID™, BLACKBERRY® OS, or the like.

[027] In some embodiments, the computer system 300 may implement a web browser 308 stored program component. The web browser 308 may be a hypertext viewing application, such as Microsoft Internet Explorer, Google Chrome, Mozilla Firefox, Apple Safari, etc. Secure web browsing may be provided using Hypertext Transport Protocol Secure (HTTPS), Secure Sockets Layer (SSL), Transport Layer Security (TLS), etc. Web browsers 308 may utilize facilities such as AJAX, DHTML, Adobe Flash, JavaScript, Java, Application Programming Interfaces (APIs), etc. In some embodiments, the computer system 300 may implement a mail server stored program component. The mail server may be an Internet mail server such as Microsoft Exchange, or the like. The mail server may utilize facilities such as ASP, ActiveX, ANSI C++/C#, Microsoft .NET, Common Gateway Interface (CGI) scripts, Java, JavaScript, PERL, PHP, Python, WebObjects, etc. The mail server may utilize communication protocols such as Internet Message Access Protocol (IMAP), Messaging Application Programming Interface (MAPI), Microsoft Exchange, Post Office Protocol (POP), Simple Mail Transfer Protocol (SMTP), or the like. In some embodiments, the computer system 300 may implement a mail client stored program component. The mail client may be a

mail viewing application, such as Apple Mail, Microsoft Entourage, Microsoft Outlook, Mozilla Thunderbird, etc.

[028] Embodiments of the present disclosure discloses a container management system and method for automating the process of building container cluster without manual intervention.

[029] Embodiments of the present disclosure reduces the time consumption for building container cluster.

[030] Furthermore, one or more computer-readable storage media may be utilized in implementing embodiments consistent with the present disclosure. A computer-readable storage medium refers to any type of physical memory on which information or data readable by a processor may be stored. Thus, a computer-readable storage medium may store instructions for execution by one or more processors, including instructions for causing the processor(s) to perform steps or stages consistent with the embodiments described herein. The term "computer-readable medium" should be understood to include tangible items and exclude carrier waves and transient signals, i.e., be non-transitory. Examples include Random Access Memory (RAM), Read-Only Memory (ROM), volatile memory, non-volatile memory, hard drives, Compact Disc (CD) ROMs, DVDs, flash drives, disks, and any other known physical storage media.
media.

[031] The described operations may be implemented as a method, system or article of manufacture using standard programming and/or engineering techniques to produce software, firmware, hardware, or any combination thereof. The described operations may be implemented as code maintained in a "non-transitory computer readable medium," where a processor may read and execute the code from the computer readable medium. The processor is at least one of a microprocessor and a processor capable of processing and executing the queries. A non-transitory computer readable medium may include media such as magnetic storage medium (e.g., hard disk drives, floppy disks, tape, etc.), optical storage (CD-ROMs, DVDs, optical disks, etc.), volatile and non-volatile memory devices (e.g., EEPROMs, ROMs, PROMs, RAMs, DRAMs, SRAMs, Flash Memory, firmware, programmable logic, etc.), etc. Further, non-transitory computer-readable media may include all computer-readable media except for a transitory. The code implementing the described operations may further be

implemented in hardware logic (e.g., an integrated circuit chip, Programmable Gate Array (PGA), Application Specific Integrated Circuit (ASIC), etc.).

[032] The illustrated steps are set out to explain the exemplary embodiments shown, and it should be anticipated that ongoing technological development will change the manner in which particular functions are performed. These examples are presented herein for purposes of illustration, and not limitation. Further, the boundaries of the functional building blocks have been arbitrarily defined herein for the convenience of the description. Alternative boundaries can be defined so long as the specified functions and relationships thereof are appropriately performed. Alternatives (including equivalents, extensions, variations, deviations, etc., of those described herein) will be apparent to persons skilled in the relevant art(s) based on the teachings contained herein. Such alternatives fall within the scope and spirit of the disclosed embodiments. Also, the words "comprising," "having," "containing," and "including," and other similar forms are intended to be equivalent in meaning and be open ended in that an item or items following any one of these words is not meant to be an exhaustive listing of such item or items or meant to be limited to only the listed item or items. It must also be noted that as used herein and in the appended claims, the singular forms “a,” “an,” and “the” include plural references unless the context clearly dictates otherwise.

[033] Furthermore, one or more computer-readable storage media may be utilized in implementing embodiments consistent with the present disclosure. A computer readable storage medium refers to any type of physical memory on which information or data readable by a processor may be stored. Thus, a computer readable storage medium may store instructions for execution by one or more processors, including instructions for causing the processor(s) to perform steps or stages consistent with the embodiments described herein. The term “computer readable medium” should be understood to include tangible items and exclude carrier waves and transient signals, i.e., are non-transitory. Examples include random access memory (RAM), read-only memory (ROM), volatile memory, non-volatile memory, hard drives, CD ROMs, DVDs, flash drives, disks, and any other known physical storage media.

[034] Finally, the language used in the specification has been principally selected for readability and instructional purposes, and it may not have been selected to delineate or

circumscribe the inventive subject matter. Accordingly, the disclosure of the embodiments of the disclosure is intended to be illustrative, but not limiting, of the scope of the disclosure.

[035] With respect to the use of substantially any plural and/or singular terms herein, those having skill in the art can translate from the plural to the singular and/or from the singular to the plural as is appropriate to the context and/or application. The various singular/plural permutations may be expressly set forth herein for sake of clarity.

SYSTEM AND METHOD FOR BUILDING CONTAINER CLUSTER

ABSTRACT

The present disclosure provides a method and container management system for building container cluster. The container management system based on user requirement may automate hardening of operating system and configure network and storage for adding new container. The container management system obtains and configures load balancer VIP without any manual intervention. Thereafter, the container management system performs end-to-end validation by deploying test application and validating the test application. Thus, the present disclosure reduces time consumption for building container clusters based on the user requirement.

1/3

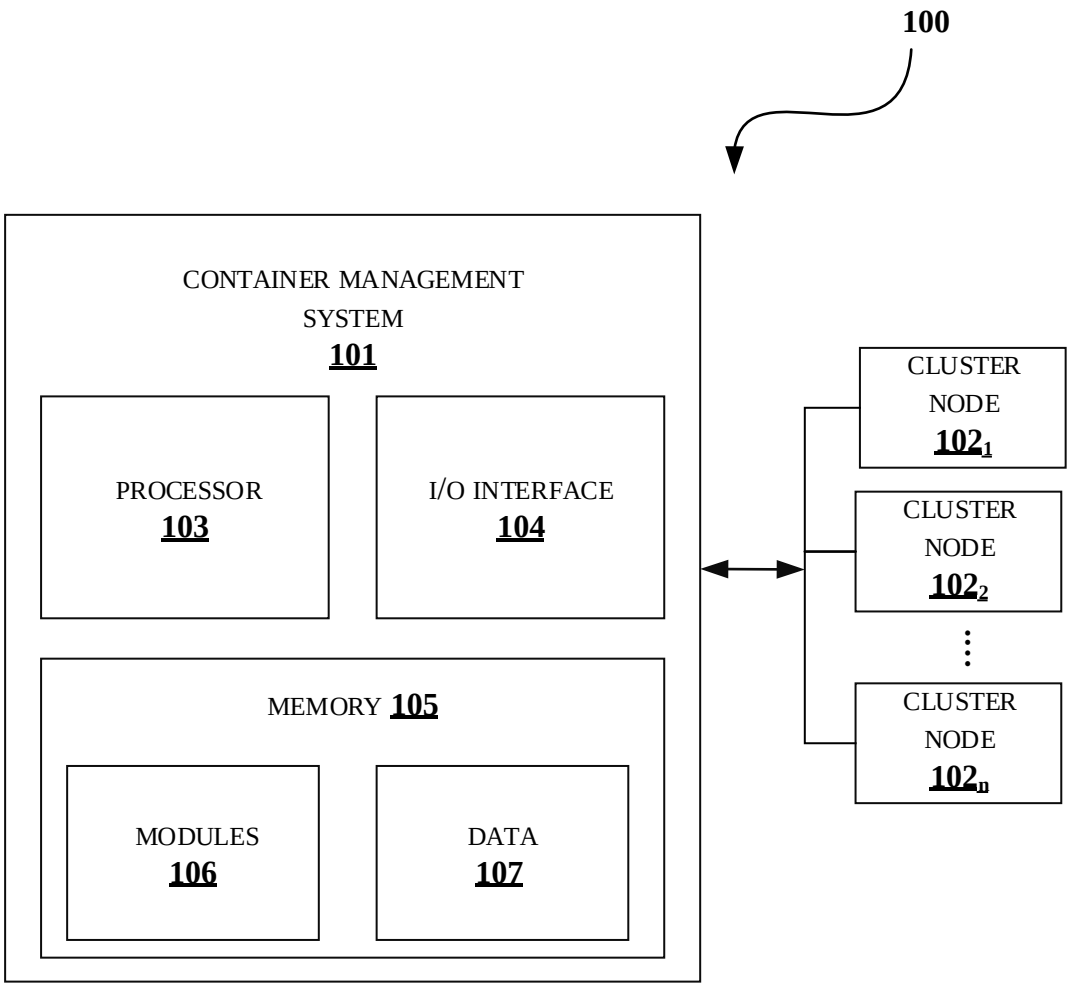
100
CONTAINER MANAGEMENT SYSTEM
101
PROCESSOR
103
I/O INTERFACE
104
MEMORY 105
MODULES
106
DATA
107
CLUSTER NODE
102₁
CLUSTER NODE
102₂
CLUSTER NODE
102ₙ

Figure 1

2/3

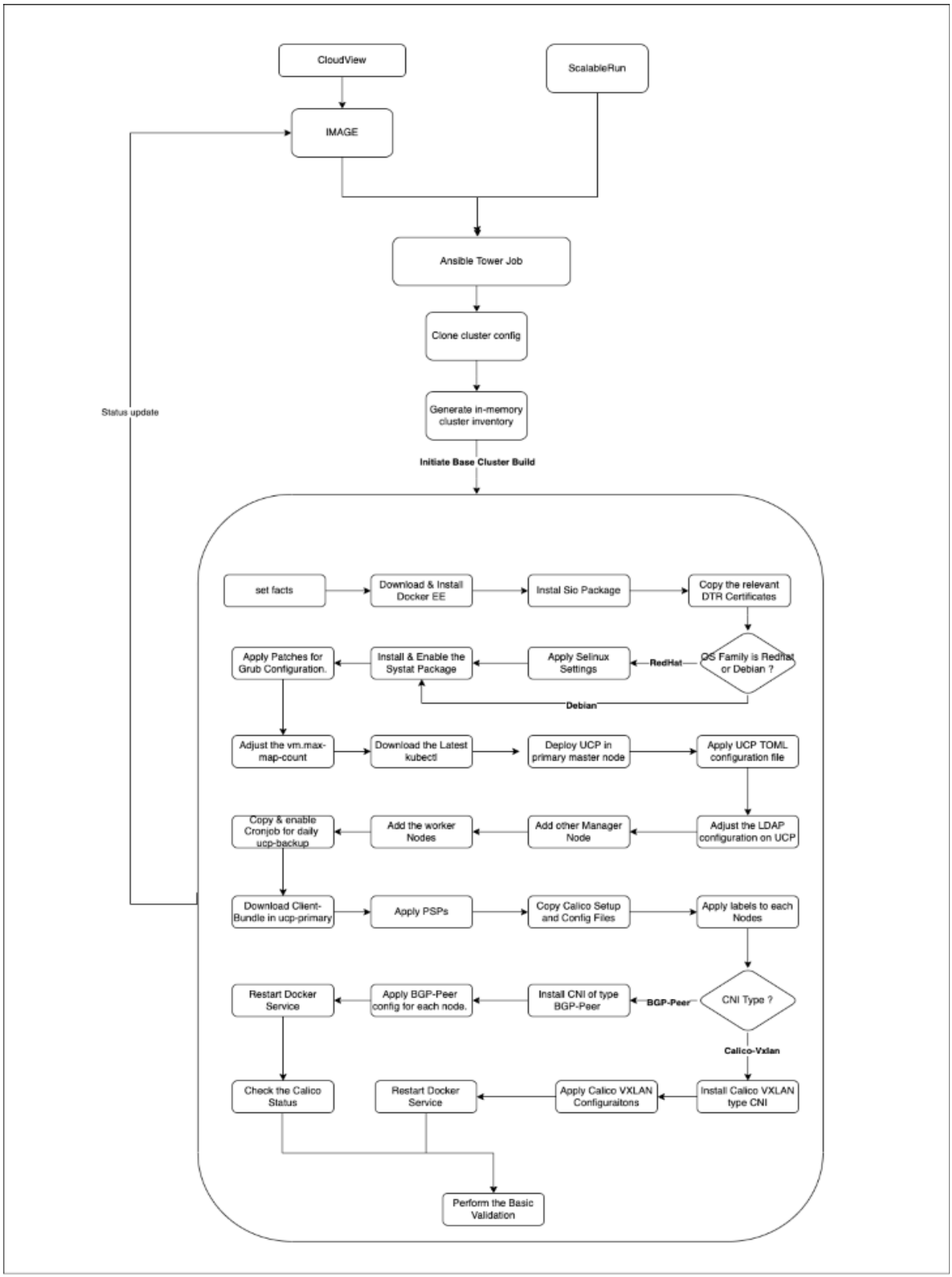
CloudView
ScalableRun
IMAGE
Ansible Tower Job
Clone cluster config
Generate in-memory cluster inventory
Status update
Initiate Base Cluster Build
set facts
Download & Install Docker EE
Instal Sic Package
Copy the relevant DTR Certificates
OS Family is Redhat or Debian ?
RedHat
Debian
Apply Selinux Settings
Install & Enable the Systat Package
Apply Patches for Grub Configuration.
Adjust the vm.max-map-count
Download the Latest kubectl
Deploy UCP in primary master node
Apply UCP TOML configuration file
Adjust the LDAP configuration on UCP
Add other Manager Node
Add the worker Nodes
Copy & enable Cronjob for daily ucp-backup
Download Client-Bundle in ucp-primary
Apply PSPs
Copy Calico Setup and Config Files
Apply labels to each Nodes
CNI Type ?
BGP-Peer
Calico-Vxlan
Install CNI of type BGP-Peer
Apply BGP-Peer config for each node.
Restart Docker Service
Install Calico VXLAN type CNI
Apply Calico VXLAN Configuraitons
Restart Docker Service
Check the Calico Status
Perform the Basic Validation

Figure 2

3/3

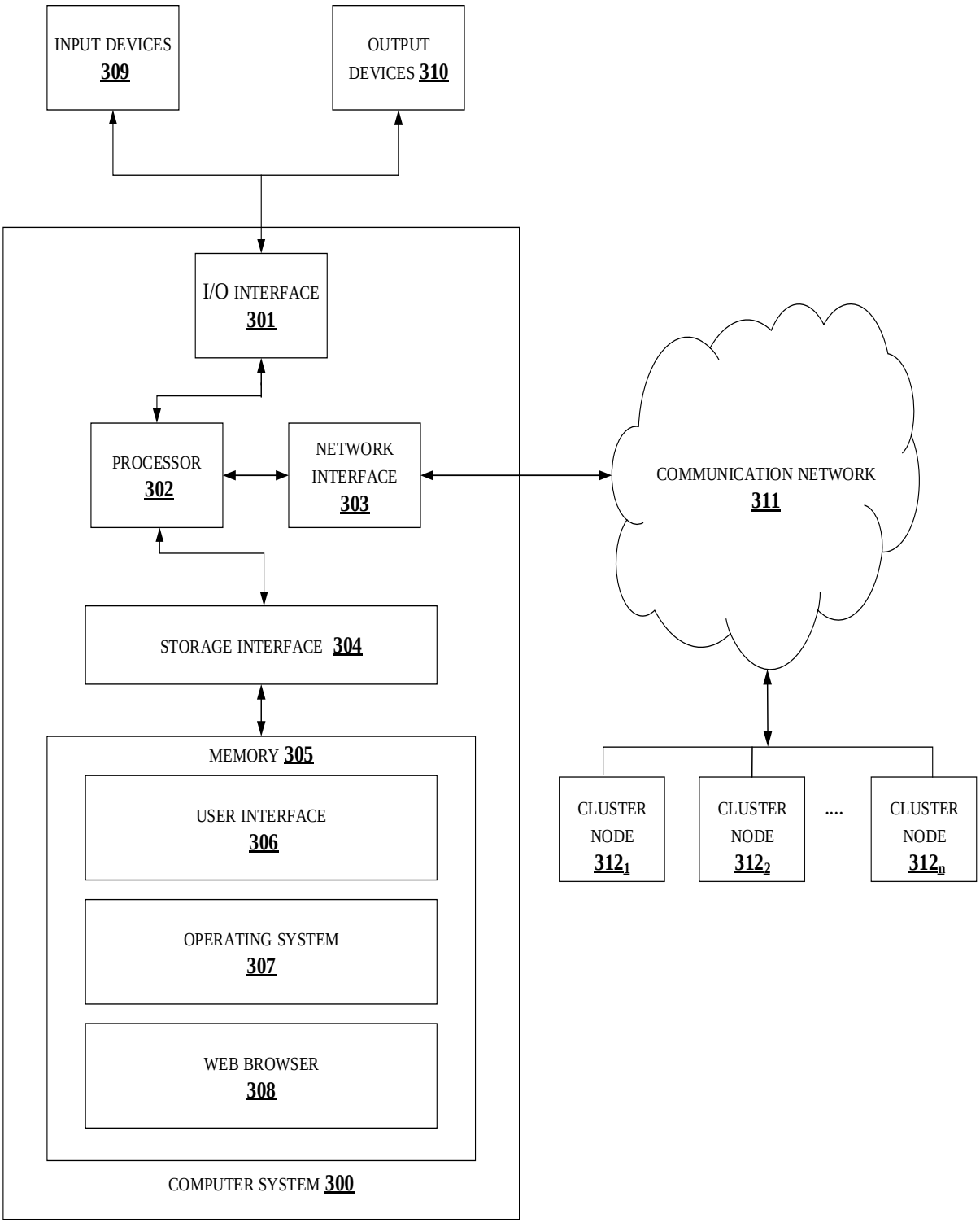
INPUT DEVICES
309
OUTPUT
DEVICES 310
I/O INTERFACE
301
PROCESSOR
302
NETWORK
INTERFACE
303
COMMUNICATION NETWORK
311
STORAGE INTERFACE 304
MEMORY 305
USER INTERFACE
306
OPERATING SYSTEM
307
WEB BROWSER
308
COMPUTER SYSTEM 300
CLUSTER
NODE
312$_1$
CLUSTER
NODE
312$_2$
....
CLUSTER
NODE
312$_n$

Figure 3