January 2023

A METHOD FOR ADDING IMAGE CLUSTER NODE APIS

Arunachalam Jayaraman
VISA

Krishna Suresh Mankuzhy
VISA

Anitha Mariajesiyan
VISA

Bharathi Sivasamy
VISA

Michael Guerin Archambeaud
VISA

See next page for additional authors

Inventor(s)
Arunachalam Jayaraman, Krishna Suresh Mankuzhy, Anitha Mariajesiyan, Bharathi Sivasamy, Michael Guerin Archambeaud, Dilshad T, Chunhui Zhong, Panneer Perumal, Kothe Pawan, and Bala Dengale

TITLE: “A METHOD FOR ADDING IMAGE CLUSTER NODE APIS”

VISA

Arunachalam Jayaraman

Krishna Suresh Mankuzhy

Anitha Mariajesiyan

Bharathi Sivasamy

Michael Guerin-Archambeaud

Dilshad T

Chunhui Zhong

Panneer Perumal

Kothe Pawan

Bala Dengale

TECHNICAL FIELD

[0001] This disclosure relates generally to the field of IT operation process. More particularly, the disclosure discloses method for adding image cluster node APIs.

BACKGROUND

[0002] One of the existing technologies discloses that an inventory consists of files where the details of all the managed nodes are disclosed. The existing technologies describe that the controller node and the inventory file's absolute path are disclosed in the Ansible configuration file (ansible. cfg) so that Ansible can recognize the controller nodes and run commands on them. Further, the existing technology discloses how Ansible helps us complete multiple steps of software installation and use a single script file, which can be implemented on multiple hosts simultaneously.

[0003] Another existing technology discloses a process of deploying service container resources for physical nodes in a cluster. A service is created first, so that the service container resources can be deployed for the physical nodes in the cluster by using the service. However, there is a problem that the container resource cannot be deployed on each physical node in the cluster quickly and on a large scale, and the container resource management for each service is inconvenient.

[0004] Further the existing technologies disclose REST APIs which initiate a node addition to an existing cluster, then updates the status of the Jenkins job to the MongoDB which may be a tedious process.

SUMMARY

[0005] According to some non-limiting embodiments, the present disclosure discloses a method for adding image cluster node APIs. The objective of the present disclosure is to initiate cluster add node to a Jenkins pipeline by calling the Jenkins APIs. The present disclosure focuses to automatically update the status of the job to the MongoDB.

[0006] In some embodiments, the present disclosure focuses on the below process:

a. The process of adding one or more nodes to the image cluster by initiating Jenkins APIs.

b. Configuring the cluster files and validating the input data related to new node.

c. Initiating capacity add playbook and updating status back to the image cluster

[0007] The present research work provides an advantage as the present disclosure uses Jenkins APIs which allows companies to speed up their software development process quite significantly. Jenkins is capable of integrating different types of software development lifecycle processes, including building, testing, deployments and making use of plugins which may be very important for ensuring continuous integration. Further, it may help developers to commit their code to Jenkins server which may automatically create a build of that code and run it through testing. If the build fails, Jenkins will notify developers of the errors so they can be fixed quickly.

[0008] These and other features and characteristics of the present invention, as well as the methods of operation and functions of the related elements of structures and the combination of parts and economies of manufacture, will become more apparent upon consideration of the following description with reference to the accompanying drawings, all of which form a part of this specification, wherein like reference numerals designate corresponding parts in the various figures. It is to be expressly understood, however, that the drawings are for the purpose of illustration and description only and are not intended as a definition of the limits of the invention. As used in the specification, the singular form of “a,” “an,” and “the” include plural referents unless the context clearly dictates otherwise.

BRIEF DESCRIPTION OF THE DRAWINGS AND APPENDICES

[0009] Additional advantages and details of non-limiting embodiments are explained in greater detail below with reference to the exemplary embodiments that are illustrated in the accompanying schematic figures, in which:

[0010] FIG.1 discloses a brief flowchart illustrating addition of one or more nodes to image cluster, according to some principles of the present disclosure;

[0011] FIG. 2 discloses a detailed flowchart illustrating configurations of the cluster files and validating the input data related to new node according to some principles of the present disclosure; and

[0012] FIG.3A-3C disclose a detailed flowchart illustrating addition of one or more nodes to image cluster, according to some principles of the present disclosure.

DESCRIPTION OF THE DISCLOSURE

[0013] 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.

[0014] 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.

[0015] 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 or system or apparatus.

[0016] 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.

[0017] The terms "including", "comprising", “having” and variations thereof mean "including but not limited to", unless expressly specified otherwise.

[0018] As used herein, the terms “communication” and “communicate” may refer to the reception, receipt, transmission, transfer, provision, and/or the like of information (e.g., data, signals, messages, instructions, commands, and/or the like). For one unit (e.g., a device, a system, a component of a device or system, combinations thereof, and/or the like) to be in communication with another unit means that the one unit is able to directly or indirectly receive information from and/or transmit information to the other unit. This may refer to a direct or indirect connection (e.g., a direct communication connection, an indirect communication connection, and/or the like) that is wired and/or wireless in nature. Additionally, two units may be in communication with each other even though the information transmitted may be modified, processed, relayed, and/or routed between the first and second unit. For example, a first unit may be in communication with a second unit even though the first unit passively receives information and does not actively transmit information to the second unit. As another example, a first unit may be in communication with a second unit if at least one intermediary unit (e.g., a third unit located between the first unit and the second unit) processes information received from the first unit and communicates the processed information to the second unit. In some non-limiting embodiments, a message may refer to a network packet (e.g., a data packet and/or the like) that includes data. It will be appreciated that numerous other arrangements are possible.

[0019] As used herein, the term “merchant” may refer to an individual or entity that provides goods and/or services, or access to goods and/or services, to customers based on a transaction, such as a payment transaction. The term “merchant” or “merchant system” may also refer to one or more computer systems operated by or on behalf of a merchant, such as a server computer executing one or more software applications. A “point-of-sale (POS) system,” as used herein, may refer to one or more computers and/or peripheral devices used by a merchant to engage in payment transactions with customers, including one or more card readers, near-field communication (NFC) receivers, RFID receivers, and/or other contactless transceivers or receivers, contact-based receivers, payment terminals, computers, servers, input devices, and/or other like devices that can be used to initiate a payment transaction.

[0020] As used herein, the term payment card may be (e.g., a credit or debit card), a gift card, a smartcard, smart media, a payroll card, a healthcare card, a wrist band, a machine-readable medium containing account information, a keychain device or fob, an RFID transponder, a retailer discount or loyalty card, a mobile device executing an electronic wallet application, a personal digital assistant, a security card, an access card, a wireless terminal, and/or a transponder, as examples.

[0021] As used herein, the term “computing device” may refer to one or more electronic devices that are configured to directly or indirectly communicate with or over one or more networks. A computing device may be a mobile or portable computing device, a desktop computer, a server, and/or the like. Furthermore, the term “computer” may refer to any computing device that includes the necessary components to receive, process, and output data, and normally includes a display, a processor, a memory, an input device, and a network interface. A “computing system” may include one or more computing devices or computers. An “application” or “Application Program Interface” (API) refers to computer code or other data sorted on a computer-readable medium that may be executed by a processor to facilitate the interaction between software components, such as a client-side front-end and/or server-side back-end for receiving data from the client. An “interface” refers to a generated display, such as one or more graphical user interfaces (GUIs) with which a user may interact, either directly or indirectly (e.g., through a keyboard, mouse, touchscreen, etc.). Further, multiple computers, e.g., servers, or other computerized devices, such as an autonomous vehicle including a vehicle computing system, directly or indirectly communicating in the network environment may constitute a “system” or a “computing system”.

[0022] It will be apparent that systems and/or methods, described herein, can be implemented in different forms of hardware, software, or a combination of hardware and software. The actual specialized control hardware or software code used to implement these systems and/or methods is not limiting of the implementations. Thus, the operation and behavior of the systems and/or methods are described herein without reference to specific software code, it being understood that software and hardware can be designed to implement the systems and/or methods based on the description herein.

[0023] **FIG.1** discloses a brief flowchart illustrating addition of one or more nodes to image cluster, according to some principles of the present disclosure.

[0024] In some embodiments, the present disclosure calls the Jenkins APIs to initiate addition of one or more nodes to the image cluster. Jenkins will initiate appropriate node-addition playbooks to accomplish one or more tasks. Generally, a Cloud View (CV) means the software-solution developed to conduct a software and hardware scan, to enter service platforms, applications, virtual entities, terminal server, and virtual desktops, to get an automatic match with corresponding products and licenses as well as maintenance of license contract or support and a tool-based support regarding Software Asset Management audits. In the present disclosure the cloud view consists of n-nodes which needs to be added to a cluster. Further, when a user wishes to add a new node to a cluster, the user may use a desired Uniform Resource Locator (URL) and may redirect to the element level automation API which in turn calls Jenkins API. The Jenkins API may allow to change configuration and automate minor tasks on nodes and jobs. For example, the user may use Jenkins to get information about recently completed builds or to get the revision number of the last successful build in order to trigger some kind of release process. In other words, Jenkins is a continuous integration tool that allows continuous development, test and deployment of newly created codes.

[0025] In some embodiments, when the Jenkins API is called, Jenkins checks for the available new nodes that needs to be added to the cluster. When the new nodes are identified it may be stored in the Ansible playbook. Generally, the Ansible Playbooks are lists of tasks that automatically execute against hosts. Groups of hosts form Ansible inventory. Each module within an Ansible Playbook performs a specific task. Each module contains metadata that determines when and where a task is executed. In other words, Ansible Playbooks offer a repeatable, re-usable, simple configuration management and multi-machine deployment system, one that is well suited to deploying complex applications. For instance, the cloud view contains one or more nodes [1..n]. When the user wishes to add 10 new nodes to the cluster, the user may be redirected to the element level automation API which may call Jenkins APIs. The Jenkins APIs may check for the available 10 new nodes that need to be added to the cluster and stores 10 new nodes in the ansible playbook as shown in **FIG.1**.

[0026] In some embodiments, a processor may check if all new nodes that are identified are successfully added to the ansible playbook- add nodes.. If all new nodes are added successfully, responses with details of success status including success step indicating the nodes that are successfully added to ansible playbook- add nodes may be sent to Jenkins APIs. In some embodiments, if any one of the new nodes fails, then a response with details of failure status including failure step indicating the nodes that failed to be added to ansible playbook- add nodes may be sent to Jenkins APIs.

[0027] In some embodiments, the process may validate the ansible playbook which contains new nodes. When there is successful node validation, then a suitable response with detailed success status of 1-10 nodes is sent to Jenkins API in which details related to the nodes that are successfully validated may be present. In some embodiments, if any of the nodes among 1-10 fail during the process of validation step, then a suitable response with details status of 1-10 nodes that have failed may be sent to Jenkins API in which details related to the nodes that are not validated are present.

[0028] **FIG. 2** discloses a detailed flowchart illustrating configuration of the cluster files and validation of the input data related to new nodes according to some principles of the present disclosure.

[0029] In some embodiments, the user or a developer who wishes to add to one or more nodes to the existing cluster may provide one or more data inputs. The one or more user inputs may be given to image cluster add node API. Further, the image cluster add node API may receive the information such as user/ developer name, the list of one or more nodes that the user wishes to add. For instance, the user/ developer may provide details such as name, designation, source code details, number of new nodes and the like. Initially, a computing system may validate the user details. If the user details are determined to be valid, then the data input provided by the user may be validated. Upon user validation and input data validation, Jenkins API may be called to check for the availability of new nodes. When one or more new nodes are identified, the one or more new nodes may be stored in Ansible which may be further stored in source-available cross-platform document-oriented database such as Mango DB. In some embodiments, if the user validation fails, then a processor associated with the computing system may throw an error which may indicate details of error code as shown in **FIG.2**.

[0030] In some embodiments, the user may provide tracking Id associated with the image cluster to add one or more new nodes to the existing cluster. Further, the image cluster add node API may receive the information such as user details and user tracking id which may be further validated. Initially, the computing system may validate the user details. If the user details are validated, then the tracking id provided by the user may be validated. If both the credentials are validated then the computing system may perform job id validation, if the job id validation is validated then the computing system may call Jenkins to perform the task which may be stored in database such as Mongo DB. In some embodiments, if any one of the validations such as user validation, tracking id validation and job id validation fails, the computing system may display the error which may also be stored in the database such as Mongo DB as shown in **FIG.2**.

[0031] FIG.3 A-C discloses a detailed flow chart illustrating addition of one or more nodes to image cluster according to some principles of the present disclosure.

[0032] In some embodiments, the cloud view capacity add consists of new node that the user or the developer wishes to add to the existing cluster. Further, all the available one or more new nodes may be stored in an ansible tower job. A job is an instance of Tower launching an Ansible playbook against an inventory of hosts. The Jobs link displays a list of jobs and their status–shown as completed successfully or failed, or as an active (running) job. Generally, all the attributes of the source cluster (except the label) are copied over to the new cluster, but the user can override any of them when creating the clone. The one or more existing clusters may be cloned to generate a single file. Further, the processor checks if the clone cluster configuration files are generated. If the file which may be formed by cloning one or more clusters is present, then the processor may generate in-memory cluster inventory to store all the generated files as shown in **FIG.3A.** If there are no clone cluster configuration files, the processor may generate files and store the generated files in the memory cluster inventory as shown in **FIG.3A.** Further, the processor may initiate capacity add play which may check for all the available new nodes that need to be added to the image cluster which is explained below in **FIG.3B**.

[0033] In some embodiments, the processor may download and install Docker Enterprise Edition. Docker EE provides a pluggable architecture for computer, networking, and storage providers, and open APIs that enable Docker EE to easily integrate into user systems. Docker EE includes Docker Universal Control Plane (UCP) for cluster management and Docker Trusted Registry (DTR) for Docker image storage.

[0034] In some embodiments, the processor may install Sio package. The Sio package SIO is a serial port driver package for OS/2. It is designed to not only improve performance over OS/2's default serial drivers, but also improve compatibility. SIO contains a virtualized FOSSIL (VX00) driver that can be loaded to provide FOSSIL support to DOS based communications software. SIO later added the ability to create virtualized COM ports, which, combined with the included program VMODEM, allows incoming telnet connections to be directed toward the virtualized COM port.

[0035] In some embodiments, the processor may copy UCP image packages and load details.

[0036] In some embodiments, the processor may copy the relevant DTR certificates and the calico setup and configure the files. Generally, once the user deploys Docker Trusted Registry (DTR), the user can use it to store your Docker images and deploy services to UCP using those images. Docker UCP integrates out of the box with Docker Trusted Registry (DTR). This means that you can deploy services from the UCP web UI, using Docker images that are stored in DTR.

[0037] In some embodiments, the selinux policy may be installed. Generally, selinux policy describes the access permissions for all users, programs, processes, and files, and for the devices upon which the user can act. The user can configure selinux to implement either Targeted Policy or Multilevel Security (MLS) Policy. Further, the processor may install and enable the system packages.

[0038] In some embodiments, the processor may apply patches for grub configurations and may adjust the vm_max_map-count.

[0039] In some embodiments, the processor may add UCP manager if node to be added is manager and add the worker nodes if the new node to be added is a worker.

[0040] In some embodiments, the processor may pause the transaction/ process for the one or more new nodes to join the existing cluster and setup calico facts as shown in **FIG.3B.**

[0041] In some embodiments, the processor may download the latest Kubectl command which may be a command line tool that may be used to run commands against Kubernetes clusters. The process may be done by authenticating with the Master Node of your cluster and making API calls to do a variety of management actions. Further, the processor may download client bundle in UCP primary and copy calico setup and configure the files. The client bundle contains a private and public key pair that authorizes user/ developer requests in UCP. It also contains utility scripts that users can use to configure Docker and kubectl client tools to communicate with UCP deployment as shown in **FIG.3C**.

[0042] In some embodiments, the processor may apply BGP peer configuration for each node. BGP peer configuration includes session Parameters. The BGP session parameters provide settings that involve establishing communication to the remote BGP neighbor. The session settings include the ASN of the BGP peer, authentication, and keepalive timers.

[0043] In some embodiments, the processor may apply calico node yami and apply node labels. Each of one or more nodes are labelled as each node may have calico node effect. Calico by default creates a BGP mesh between all nodes of the cluster and broadcasts the routes for container networks to all worker nodes. Each node is configured to act as a Layer 3 gateway for the subnet.

[0044] In some embodiments, the processor may restart the docker service and may pause playbooks for all configurations and check if there is any effect. Further, the processor may check for the calico node status.

[0045] In some embodiments, the processor may disable pod scheduling and drain the new node and deploy the sample pod in the test namespace with all possible tolerations as shown in **FIG.3C**.

[0046] In some embodiments, the processor may check if the test pod is in running state. Upon checking the status of pod, the processor may delete the deployment and the test namespace. Finally, the processor may update the status of addition of nodes to the image cluster (not shown in FIG.3C).

[0047] 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.

[0048] 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.

[0049] Any of the software components or functions described in this application, may be implemented as software code to be executed by a processor using any suitable computer language such as, for example, Java, C++ or Perl using, for example, conventional or object-oriented techniques. The software code may be stored as a series of instructions, or commands on a computer readable medium, such as a random-access memory (RAM), a read only memory (ROM), a magnetic medium such as a hard-drive or a floppy disk, or an optical medium such as a CD-ROM. Any such computer readable medium may reside on or within a single computational apparatus and may be present on or within different computational apparatuses within a system or network.

[0050] The above description is illustrative and is not restrictive. Many variations of the invention may become apparent to those skilled in the art upon review of the disclosure.

[0051] One or more features from any embodiment may be combined with one or more features of any other embodiment without departing from the scope of the invention.

[0052] A recitation of "a", "an" or "the" is intended to mean "one or more" unless specifically indicated to the contrary.

[0053] All patents, patent applications, publications, and descriptions mentioned above are herein incorporated by reference in their entirety for all purposes. None is admitted to be prior art.

[0054] Although the invention has been described in detail for the purpose of illustration based on what is currently considered to be the most practical and preferred embodiments, it is to be understood that such detail is solely for that purpose and that the invention is not limited to the disclosed embodiments, but, on the contrary, is intended to cover modifications and equivalent arrangements that are within the spirit and scope of the invention. For example, it is to be understood that the present invention contemplates that, to the extent possible, one or more features of any embodiment can be combined with one or more features of any other embodiment.

ABSTRACT

A METHOD FOR ADDING IMAGE CLUSTER NODE APIS

[0055] The present disclosure discloses a method for adding image cluster node APIs. The objective of the present disclosure is to initiate cluster add node to a Jenkins pipeline by calling the Jenkins APIs. The present disclosure focuses to automatically update the status of the job to the MongoDB. In some embodiments, the present disclosure focuses on adding one or more nodes to the image cluster by initiating Jenkins APIs. Further, the present disclosure includes configuring the cluster files and validating the input data related to new node and initiating capacity add playbook and updating status back to the image cluster.

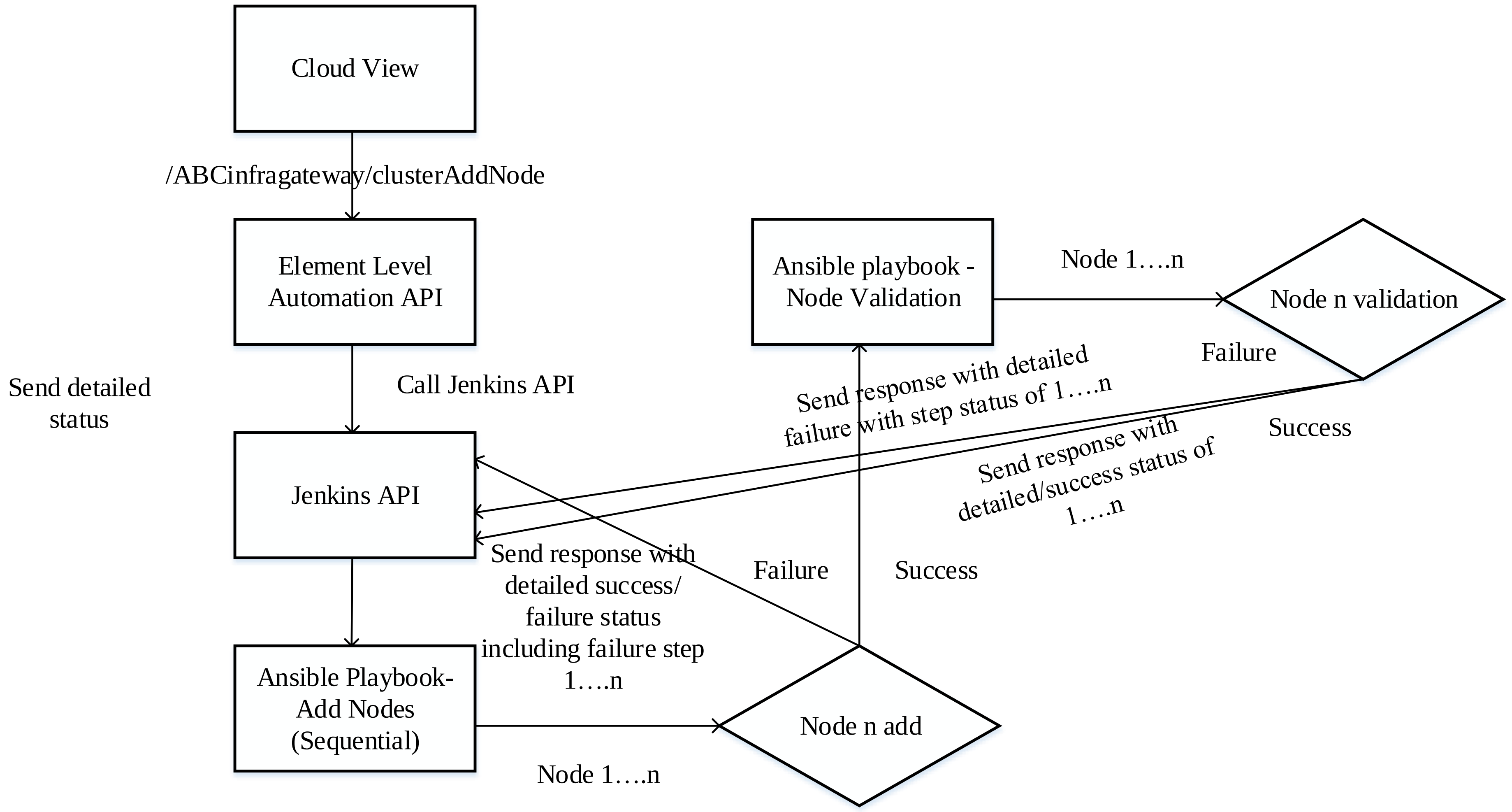
Cloud View
/ABCinfragateway/clusterAddNode
Element Level Automation API
Call Jenkins API
Send detailed status
Jenkins API
Ansible Playbook- Add Nodes (Sequential)
Node 1….n
Node n add
Send response with detailed success/ failure status including failure step 1….n
Failure
Success
Ansible playbook - Node Validation
Node 1….n
Node n validation
Failure
Send response with detailed failure with step status of 1….n
Success
Send response with detailed/success status of 1….n

FIG.1

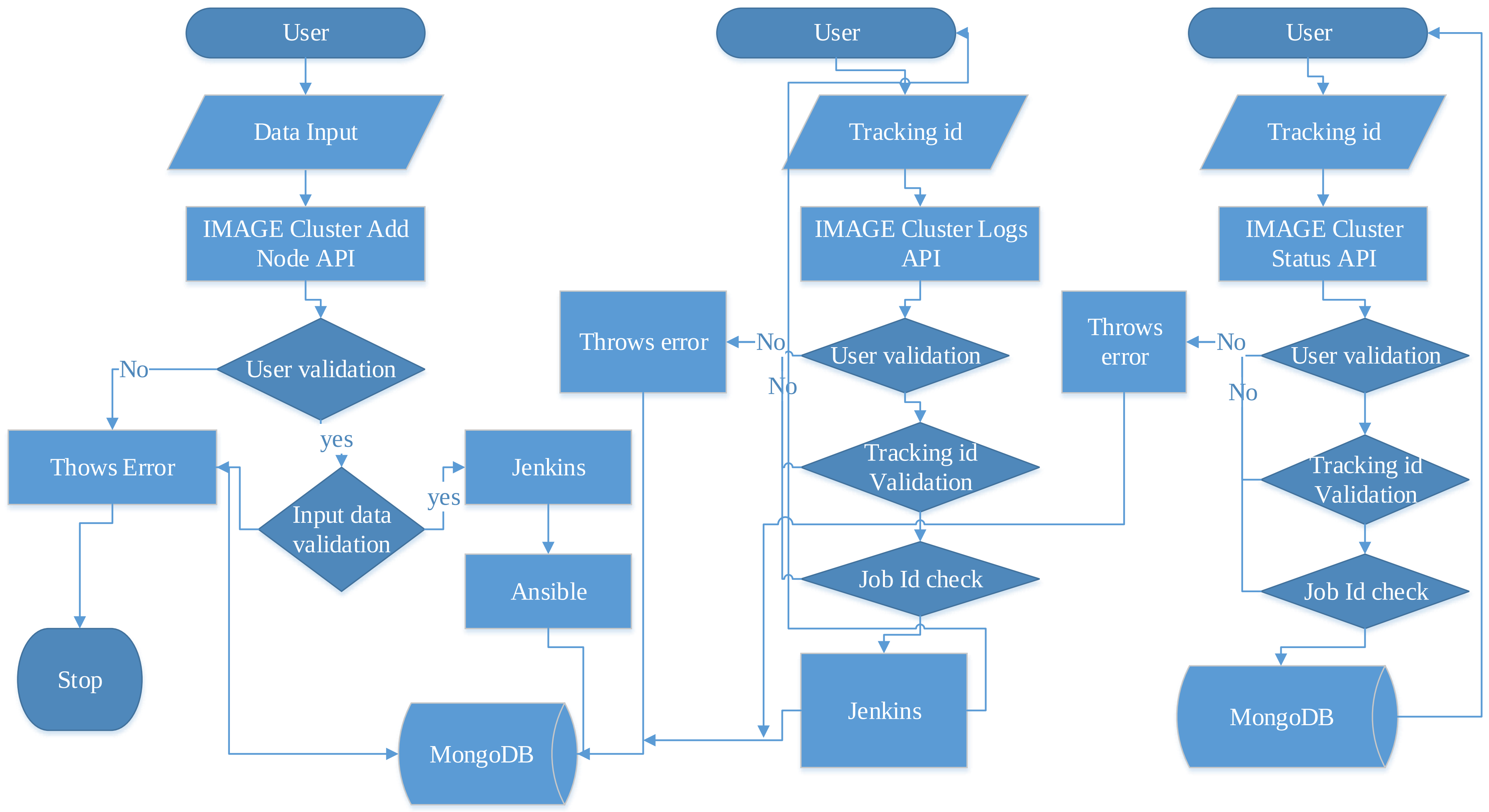
User
Data Input
IMAGE Cluster Add Node API
User validation
No
Thows Error
yes
Input data validation
yes
Jenkins
Ansible
Stop
MongoDB
User
Tracking id
IMAGE Cluster Logs API
User validation
No
Throws error
No
Tracking id Validation
Job Id check
Jenkins
User
Tracking id
IMAGE Cluster Status API
User validation
No
Throws error
No
Tracking id Validation
Job Id check
MongoDB

FIG.2

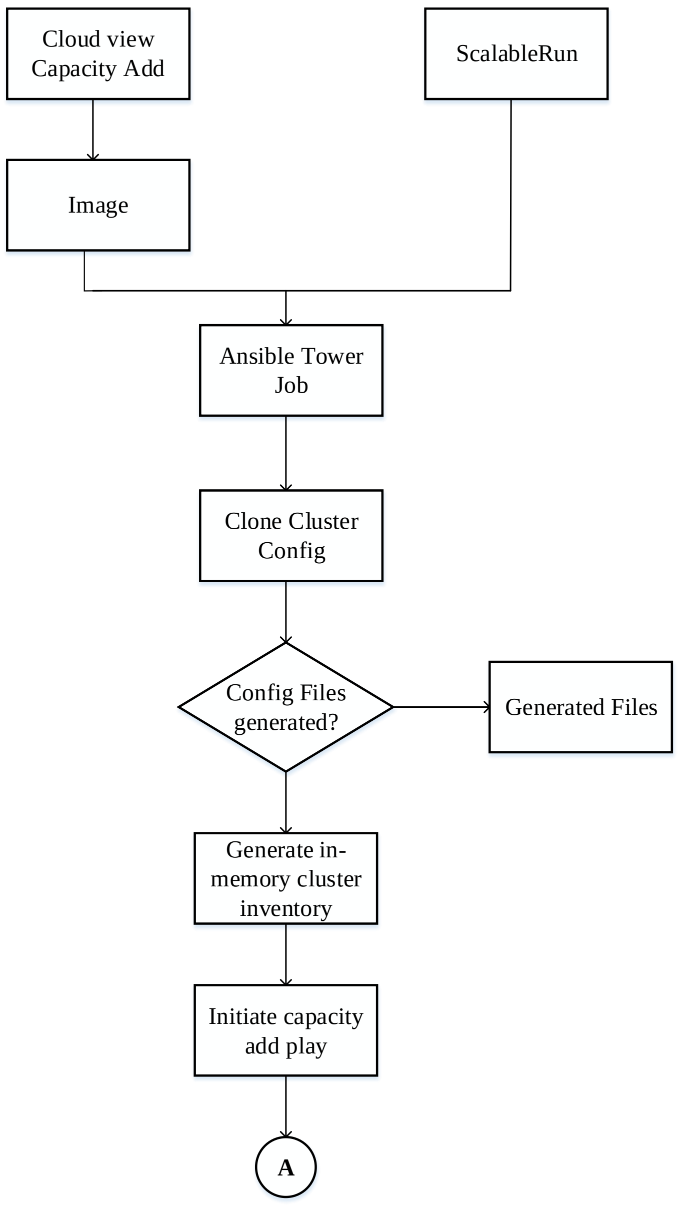
Cloud view Capacity Add
ScalableRun
Image
Ansible Tower Job
Clone Cluster Config
Config Files generated?
Generated Files
Generate in-memory cluster inventory
Initiate capacity add play
A

FIG.3A

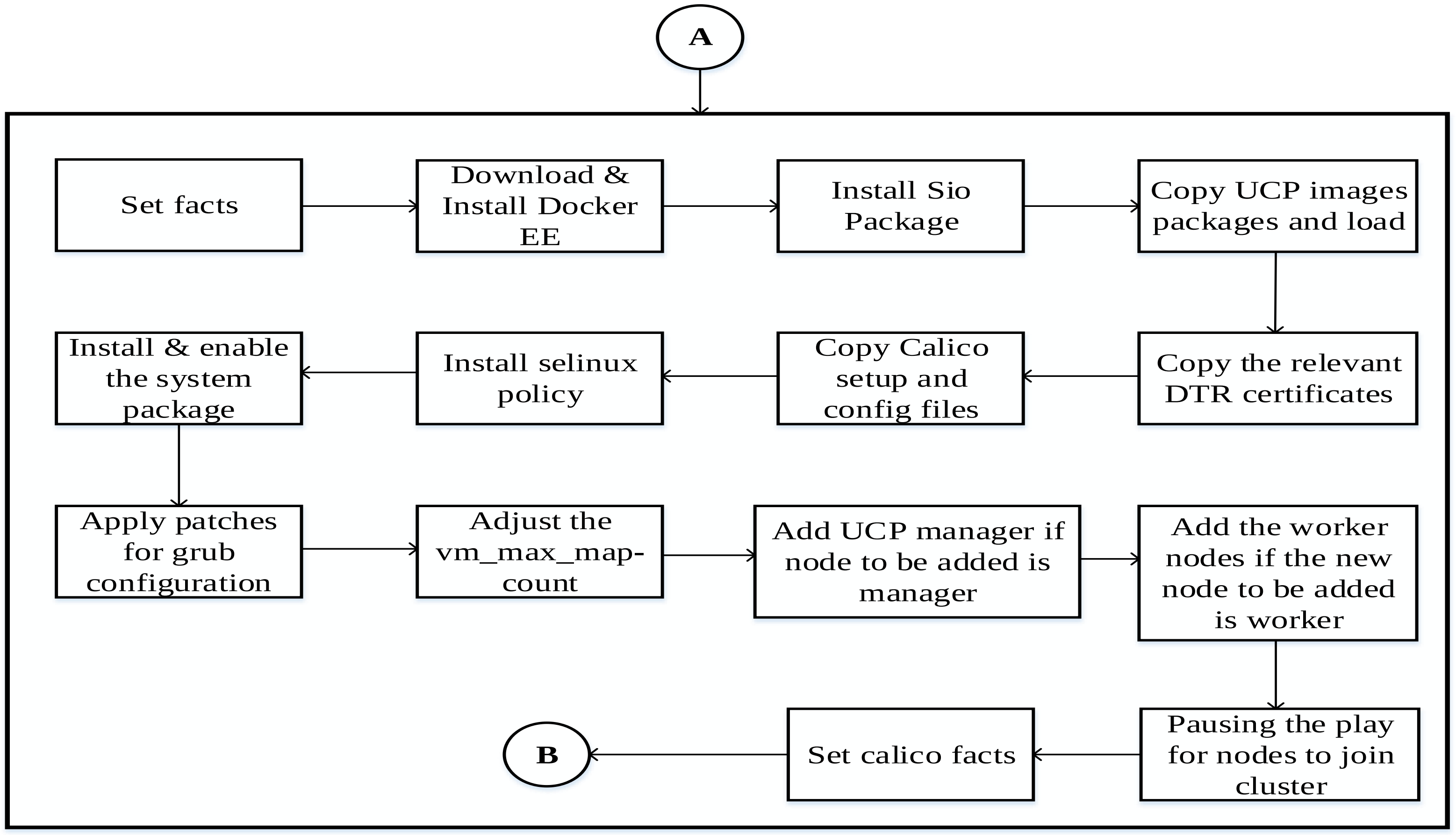
A
Set facts
Download & Install Docker EE
Install Sio Package
Copy UCP images packages and load
Copy the relevant DTR certificates
Copy Calico setup and config files
Install selinux policy
Install & enable the system package
Apply patches for grub configuration
Adjust the vm_max_map-count
Add UCP manager if node to be added is manager
Add the worker nodes if the new node to be added is worker
Pausing the play for nodes to join cluster
Set calico facts
B

FIG.3B

17

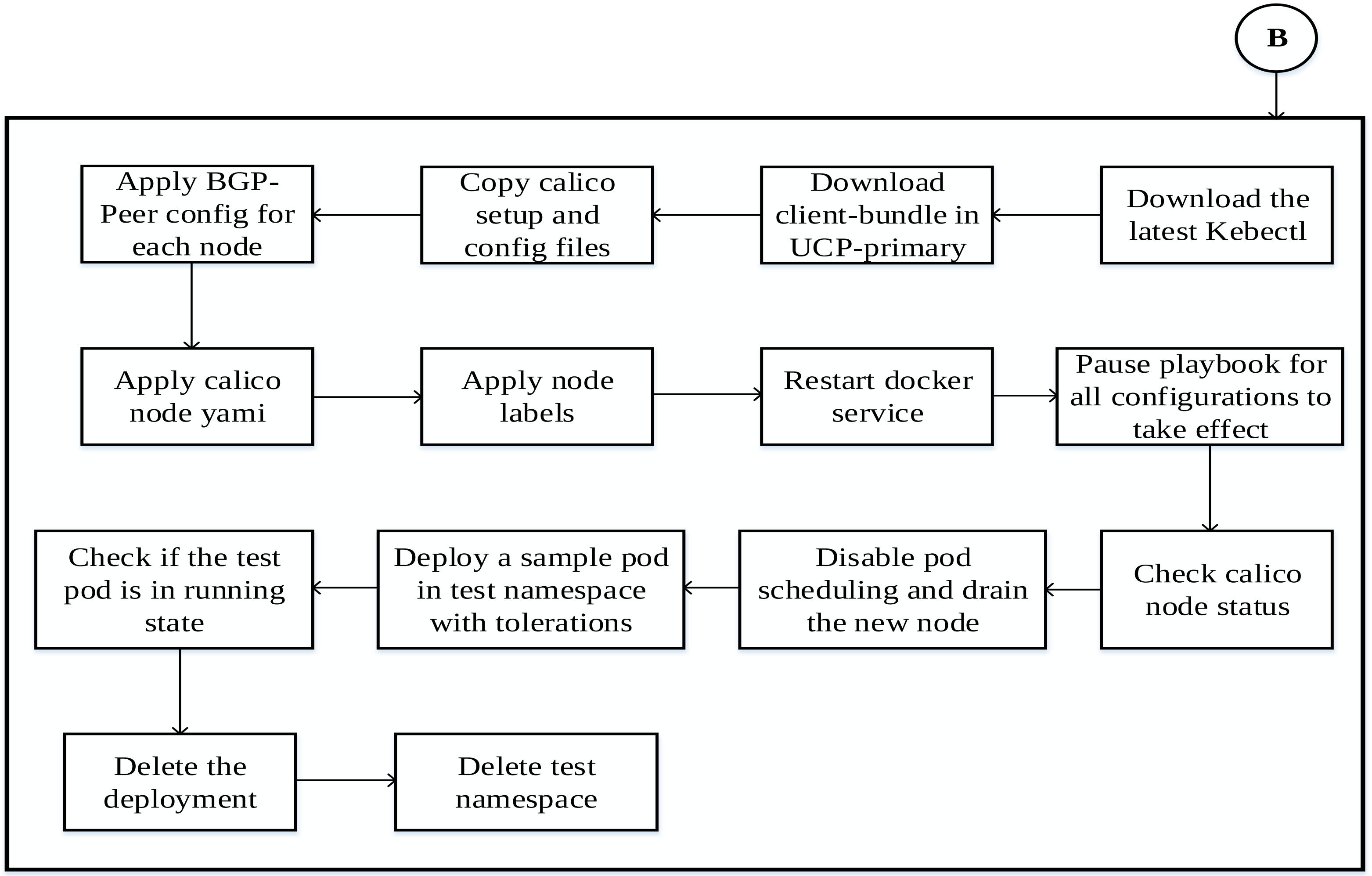
B
Download the latest Kebectl
Download client-bundle in UCP-primary
Copy calico setup and config files
Apply BGP-Peer config for each node
Apply calico node yami
Apply node labels
Restart docker service
Pause playbook for all configurations to take effect
Check calico node status
Disable pod scheduling and drain the new node
Deploy a sample pod in test namespace with tolerations
Check if the test pod is in running state
Delete the deployment
Delete test namespace

FIG.3C