



(51) International Patent Classification:

G06F 11/36 (2006.01) G06F 9/455 (2018.01)
G06F 8/76 (2018.01)

(21) International Application Number:

PCT/US2019/047482

(22) International Filing Date:

21 August 2019 (21.08.2019)

(25) Filing Language:

English

(26) Publication Language:

English

(30) Priority Data:

16/107,132 21 August 2018 (21.08.2018) US

(71) Applicant: JPMORGAN CHASE BANK, N.A. [US/US];
383 Madison Avenue, New York, New York 10179 (US).

(72) Inventors: ALEXANDER, Shawn Wesley; c/o JPMor-
gan Chase Bank, N.A., 383 Madison Avenue, New York,
New York 10179 (US). ACKROYD, Richard; c/o JPMor-
gan Chase Bank, N.A., 383 Madison Avenue, New York,
New York 10179 (US). ALAPATI, Venkata Naveen; c/o
JPMorgan Chase Bank, N.A., 383 Madison Avenue, New
York, New York 10179 (US).

(74) Agent: KING, Robert A.; c/o Greenberg Traurig, LLP, 77
West Wacker Drive, Suite 3100, Intellectual Property De-
partment, Chicago, Illinois 60601 (US).

(81) Designated States (unless otherwise indicated, for every
kind of national protection available): AE, AG, AL, AM,
AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ,
CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO,
DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN,
HR, HU, ID, IL, IN, IR, IS, JO, JP, KE, KG, KH, KN, KP,
KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME,
MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ,
OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA,
SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN,
TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States (unless otherwise indicated, for every
kind of regional protection available): ARIPO (BW, GH,
GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ,
UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ,
TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK,
EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV,
MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM,
TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW,
KM, ML, MR, NE, SN, TD, TG).

(54) Title: SYSTEMS AND METHODS FOR ON-DEMAND CONTAINER-BASED DEVELOPMENT ENVIRONMENTS

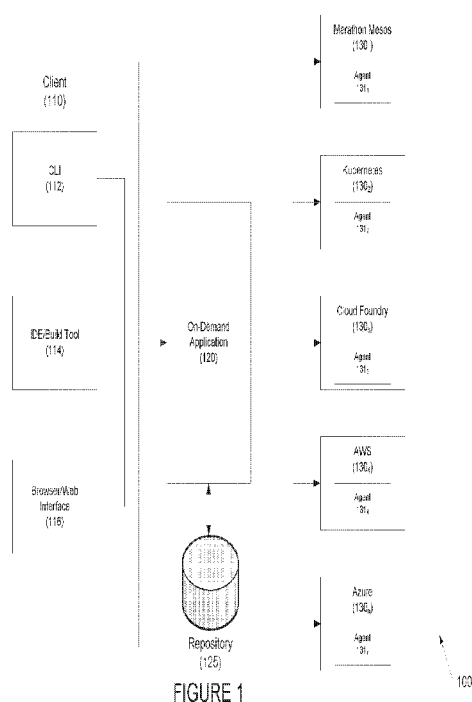


FIGURE 1

(57) Abstract: Systems and methods for on-demand container-based development environments are disclosed. According to one embodiment, in an information processing apparatus comprising at least one computer processor a method for using an on-demand development environment may include: (1) receiving, from a user interface, a request for an on-demand development environment for an application binary; (2) verifying that infrastructure for the on-demand development environment is not provisioned; (3) identifying a build template for the on-demand development environment; (4) selecting an endpoint for the on-demand development environment; (5) initiating the on-demand development environment on the endpoint using the build template; and (6) deploying the application binary to the on-demand development environment.

Published:

— *with international search report (Art. 21(3))*

SYSTEMS AND METHODS FOR ON-DEMAND CONTAINER-BASED DEVELOPMENT ENVIRONMENTS

BACKGROUND OF THE INVENTION

1. Field of the Invention

[0001] The present invention generally relates to systems and methods for on-demand container-based development environments.

2. Description Of The Related Art

[0002] Getting access to development environments is often time consuming, process-heavy, complex, and may have a long lead time. Development teams face issues where they are developing across feature branches, but must merge all their code together to run into development environments. Developers often compete for limited resources, and obtaining access to additional resources is complicated, lengthy, and time-consuming. In order to innovate, developers need access to compute on demand without overhead or bureaucracy that occurs in today's environments.

SUMMARY OF THE INVENTION

[0003] Systems and methods for on-demand container-based development environments are disclosed. According to one embodiment, in an information processing apparatus comprising at least one computer processor a method for using an on-demand development environment may include: (1) receiving, from a user interface, a request for an on-demand development environment for an application binary; (2) verifying that infrastructure for the on-demand development environment is not

provisioned; (3) identifying a build template for the on-demand development environment; (4) selecting an endpoint for the on-demand development environment; (5) initiating the on-demand development environment on the endpoint using the build template; and (6) deploying the application binary to the on-demand development environment.

[0004] In one embodiment, the method may further include determining whether a build template exists.

[0005] In one embodiment, the on-demand development environment may be a container-based development environment.

[0006] In one embodiment, the request may include at least one of a template, a source, a run command, and a port.

[0007] In one embodiment, the endpoint may be selected based on at least one functional requirement. The function requirements may include an identification one or more external services required for the application binary or a template received by the user interface.

[0008] In one embodiment, the endpoint may be selected based on at least one non-functional requirement. The non-functional requirement may include at least one of a size of the on-demand development environment, an available capacity for the endpoint, a performance of the endpoint, a data sensitivity of the application binary, and a cost of using the endpoint.

[0009] In one embodiment, the build template may include at least one of an identification of an operating system, an application infrastructure, an on-demand agent, a port, and debugging tool.

[0010] In one embodiment, the build template may include a Docker image, and the Docker image may be retrieved from an image repository.

[0011] In one embodiment, the build template may be selected based on a build template used in a prior request for an on-demand development environment.

[0012] In one embodiment, the endpoint may be Marathon Mesos, Cloud Foundry, AWS, or Kubernetes.

[0013] In one embodiment, the method may further include deactivating the on-demand development environment.

[0014] In one embodiment, the method may further include saving a snapshot of state of the on-demand development environment.

[0015] In one embodiment, the method may further include collecting performance metrics about the on-demand development environment, wherein the performance metrics include at least one of a time for the on-demand development environment to go live, a performance of the on-demand development environment, and an available capacity of the on-demand development environment.

[0016] According to another embodiment, an on-demand development system may include a plurality of clients; a server comprising at least one computer processor executing an on-demand application; a template repository comprising a plurality of build templates; and a plurality of endpoints, each end point executing an on-demand agent. The on-demand application may receive a request for an on-demand development environment for an application binary from one of the clients; verify that infrastructure for the on-demand development environment is not provisioned; identify a build template for the on-demand development environment; retrieve the build template from the template repository; select one of the plurality of endpoints for the on-demand development

environment; initiates the on-demand development environment on the endpoint using the build template; and deploys the application binary to the on-demand development environment. The agent for the selected endpoint may unpack the application binary to the on-demand development environment.

[0017] In one embodiment, the on-demand development environment may be a container-based development environment.

[0018] In one embodiment, the endpoint may be selected based on at least one functional requirement including at least one of an identification one or more external services required for the application binary or a template received by the user interface.

[0019] In one embodiment, the endpoint may be selected based on at least one non-functional requirement including at least one of a size of the on-demand development environment, an available capacity for the endpoint, a performance of the endpoint, a data sensitivity of the application binary, and a cost of using the endpoint.

[0020] In one embodiment, the build template may include a Docker image.

BRIEF DESCRIPTION OF THE DRAWINGS

[0021] For a more complete understanding of the present invention, the objects and advantages thereof, reference is now made to the following descriptions taken in connection with the accompanying drawings in which:

[0022] Figure 1 depicts an architecture for an on-demand container-based development environment according to one embodiment; and

[0023] Figure 2 depicts method for using an on-demand container-based development environment according to one embodiment.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

[0024] Embodiments disclosed herein are directed to systems and methods for on-demand container-based development environments.

[0025] Current containers and Containers as a Service (CaaS) offerings provide virtualized servers for developers, but at significant overhead. Developers must therefore understand the internal workings of the APIs for the cloud offerings (e.g., Marathon/Mesos APIs, Kubernetes APIs, AWS APIs, etc.). Each of these express infrastructure as code to the developers in different languages. For example, each of the cloud providers may have their own API language that they have built for development teams to integrate with. Therefore, every developer has to learn, for example, the Kubernetes API, Marathon API, Cloud Foundry API, AWS APIs, etc. This can feel like completely different languages (or APIs) to a developer to memorize for each endpoint.

[0026] This causes two issues for developers. First, they have understand and code their infrastructure (additional overhead and code required) and second, developers are forced to understand competing API structures and dependencies in a very rapidly growing area.

[0027] In contrast, embodiments abstract developers from the infrastructure as code requirements by leveraging pre-built “Docker” templates. The templates cause an agent to perform application code deploy, which creates a separation of responsibilities for developer: the infrastructure as code stores the application infrastructure images (such as

OS, application tech stack), and, at runtime, the application binaries themselves are injected into the running Docker image.

[0028] Application developers then can focus on their application logic and inject it into the runtime container when required. This also creates a community of pre-defined templates for developers to leverage, which promotes re-use across our feature teams.

[0029] By segregating the application logic from the application infrastructure, developers may stand up their own instances. For example, developers may rapidly provision, or destroy, an individual container across container endpoints. The environments may run on any container platforms, including containers that are internal to the organization, and containers that are external to the organization. The developer does not need to know how to deploy on each environments, or have to specify which environments that the application workload needs to run on. Instead, the developer may define the type of environment needed, and request a new one to be provisioned and configured.

[0030] Embodiments described herein are directed to a real-time development environment for sandbox testing or branch testing with access to individual environments available for feature teams. Embodiments may provision, configure, and start disposable development and sandbox environments on demand. Base templates may be instantiated for common application architectures (e.g., Java server-less, Java tomcat, Python, go, etc.), while development teams may make their own custom templates for more complex use cases or simply provide Dockerfiles and configurations for the appropriate endpoint.

[0031] Embodiments may create an abstraction between application developers, and needing to manage an ever-growing list of Platform as a Service (PaaS) and Infrastructure as a Service (IaaS) solution APIs that can change frequently and are specific to each endpoint. Application developers may request an environment with the services and a technical stack they need without needing to configure HA Proxies, providing API/CLI syntax per endpoint (e.g., Marathon Mesos, Cloud Foundry, Kubernetes, Fargate, EC2), etc.

[0032] Embodiments may provide some or all of the following features and benefits: (1) endpoint agnostic application builds and configuration; (2) reduced lead time for environments from months to minutes; (3) development environments may be provisioned only as needed; (4) abstraction layer over ever changing public/private cloud options; (5) deploy to running containers instead of recreating container images and rebuilding environment from scratch; (6) avoid expensive upload of application binaries to a repository (e.g., Quay) and instead leverage templates and deploy application binaries to running containers; (7) cross container portability for application logic without re-configuration or re-programming; (8) the ability to fail over workloads in a heterogeneous deployment footprint (across different container providers and internal/public environments); (9) separation of application logic from infrastructure as code to create reusable templates that a community of developers can source environments from and contribute to; (10) provide “hot” deployment of application logic to running containers without having to restart containers for application changes; (11) provide individualized environments for developers for a fail fast safe playing ground; (12) provide Environment as a Service capability (versus PaaS, IaaS or CaaS solutions) in which environments are consumable by

developers without knowledge of or care about the underlying application or server infrastructure or containers; (13) provide true environment portability in that environments will span existing physical and virtual infrastructure in addition to container and cloud infrastructure; etc.

[0033] Referring to Figure 1, an architecture for an on-demand container-based development environment is disclosed according to one embodiment. In one embodiment, system 100 may include client 110, on-demand application or server 120, and endpoints 130₁, 130₂, 130₃, 130₄, and 130_n.

[0034] In one embodiment, client 110 may be used to provision and deploy environments. The environments may be for an individual, a feature branch, an application, etc. and include a single endpoint 130, multiple endpoints 130 (e.g., different types of endpoints 130, etc.).

[0035] Client 110 may include CLI 112, IDE/Build Tool 114, and browser/web interface 116. In one embodiment, developers may be presented with various, different ways to interact with on-demand application or server 120 so they may pick the one that is easiest for them to memorize. CLI 112 is a Windows and Linux compiled executable to which users may provide commands (e.g., ondemand start myapplication) from a command line window. The command window may run as a bash window for Linux environments or Windows command environment.

[0036] IDE and Build Tool 114 may provide similar functionality. With IDE and Build Tool 114, a command prompt may be provide in the existing tooling the developers use on a day-to-day basis to provide the same interaction that CLI 112 has, but with fewer clicks and in the program that they spend the majority of their time in already. As a build tool,

environments may be provisioned and started from a Continuous Integration/Continuous Delivery pipeline as a build step (e.g., download code, compile, unit test, ondemand provision environment, deploy, automated testing, ondemand delete environment, deploy to QA).

[0037] Browser/Web Interface 116 may provide an interface via, for example, a website where developers may define their application metadata, upload their application binary via a graphical user interface, etc. This may be an alternative to using CLI 112 or IDE/Build Tool 114. The developer may manually enter the information they care about and point and click to deploy.

[0038] On-demand application or server 120 may comprise a computer program or application executed by a computer processor (e.g., a server) that may create an abstraction layer across PaaS and IaaS providers. On-demand application or server 120 may determine at runtime what endpoint/provider 130 is most suitable to the binary, instantiates one or more endpoint(s), and configures a technology stack required by the application.

[0039] In one embodiment, the decision of on-demand application or server 120 may be overridden if a specific endpoint 130 is desired.

[0040] In one embodiment, on-demand application or server 120 may include a plurality of build templates, which may be configured container/server images for common environment needs (e.g., Red Hat Linux 7, Java 8 plus Tomcat, etc.). In one embodiment, build templates may be pre-built and may be stored within repository 125 (commonly called a Quay) to be instantiated as required. Thus, for common use cases, a user may ignore the underlying endpoint configuration and deployment details.

[0041] Endpoints 130₁, 130₂, 130₃, 130₄ ... 130_n may be provided. In one embodiment, each endpoint 130 may be a different container provider where multiple application templates can be started. An example endpoint 130 is Amazon Web Services (AWS). In AWS, multiple instances of the Java 8 template may be created, and each instance may be a runtime container for a single application binary. Each instance of a Java 8 template on AWS may have different configuration parameters for, for example, the size of memory, virtual CPUs, the amount of disk that is provisioned, etc.

[0042] Each instance of the template on endpoint 130 may contain one application binary and one copy of agent 131 running. Developers may also create multiple copies of an instance of a template on endpoint 130 (e.g., three Java 8 templates running on AWS). This gives the developers maximum flexibility in either very specifically defining their runtime environment (e.g., number of instances, type of template, endpoint, etc.), or giving very high level instructions and allowing on-demand application or server 120 to pick the most suitable values for the non-defined parameters.

[0043] In one embodiment, each endpoint 130 may have agent 131 installed. For example, agent 131 may be bundled with container templates or otherwise installed by on-demand application or server 120. When started (e.g., upon provisioning of a container), agent 131 may listen for commands to deploy code packages on the endpoint. Code packages may then be uploaded directly to endpoint 130 via agent 131, which may manage the installation, including stopping existing processes, deploying the new binaries to the correct locations on the filesystem, configuring runtime parameters or environment variables and starting the application itself.

[0044] Referring to Figure 2, a method for using an on-demand container-based development environment is disclosed according to one embodiment.

[0045] In step 205, a developer, at a client, may request an on-demand container-based development environment for code from a computer application, such as an on-demand application or server. In one embodiment, the developer may specify details for the environment; for example, components for the environment, such as the template, source, run command, ports, etc.

[0046] In step 210, the on-demand application or server may check to see if infrastructure for the development environment is provisioned. If it is, in step 215, the developer may be informed, and the client may communicate the code to the provisioned environment.

[0047] In one embodiment, if infrastructure exists, the endpoint having that infrastructure may be selected provided that the endpoint is still available and has capacity. In one embodiment, a user may specify, or override, the endpoint selection to force an environment to be built on an endpoint of their choice (e.g., if the developer decides they will go to production on a particular container such as AWS, there's no point in putting their development on Kubernetes).

[0048] If the user does not specify or force an endpoint, the endpoint may be selected based on, for example, functional and non-functional requirements. Examples of functional requirements include the external services required for the application binary, the type of template that they selected (e.g. some templates will only run on certain endpoints), etc. Examples of non-functional requirements include the size of instance

requested and available capacity on the endpoint (e.g., is an endpoint over 80% utilized), endpoint availability (e.g., is AWS down), performance of the endpoint (e.g., has Cloud Foundry started to slow down), data sensitivity of the application (e.g., private data cannot go to public cloud endpoints), the cost of the workload on the endpoints (where is it cheapest to run), etc.

[0049] If the infrastructure is not provisioned, in step 220, the on-demand server may identify a build template that is necessary for the environment, and may select an endpoint (as described above). In one embodiment, the build template today may include, for example, the operating system version (e.g., Red Hat Enterprise Linux 7, etc.), any combination of application infrastructure components (e.g., Java 8, Apache Tomcat 7, Python miniconda 3, etc.). A template may have multiple application infrastructure components (such as Java 8 and Tomcat 7). A template may also include a copy of the on-demand agent running, for example, on port 8090 for the application container. Debugging may be supported by including or providing monitoring and alerting tools for the application container (e.g., top, vmstat, SystemTap).

[0050] For example, the on-demand server may base the build template on specifications provided by the developer, by analyzing the job, by comparing the job to a prior job, etc.

[0051] In step 225, the on-demand server may determine whether the build template exists. In one embodiment, the on-demand application or server may have the coordinates for template(s) that have been pre-built into Docker image(s). The Docker image may contain, for example, an operating system, an application infrastructure components, and an on-demand agent).

[0052] If a build template does not exist, in step 230, the build template is created. For example, the on-demand server or application will wait until the Docker compilation is completed, and the Docker image is in the image repository before continuing. Once the Docker image is available in the image repository (either because it is being reused from a previous build or the build finishes), the on-demand server or application may instruct the endpoint provider to create a new instance of the Docker image running on that endpoint.

[0053] If the build template exists, in step 240, the on-demand server may select that build template.

[0054] In step 245, the on-demand server may initiate an instance of the development environment on the endpoint. For example, the on-demand server may provision the infrastructure within the appropriate environment (e.g., Marathon Mesos, Cloud Foundry, AWS, Kubernetes, etc.) and may start the agent on the endpoint(s).

[0055] In step 250, the on-demand server may direct the client to the agent, and the client may deploy a code package (e.g., binaries) to the agent.

[0056] In step 255, the agent may unpack the code package, and may start the application on the endpoint(s)

[0057] Following use of the environment, the environment may be deactivated. For example, a snapshot of the data on disk may be taken so that the “state” of the environment can be restored in the future if, for example, the user wants to “go back” to the environment at a certain point in the past. The API for the endpoint the environment was provisioned to may be deleted or destroyed, which also destroys the container.

[0058] In one embodiment, automated performance metrics about the time to live, the performance of the environment, and the available capacity may be captured to evaluate non-functional availability and performance of the environments. In one embodiment, these and other metrics may be used in selecting the endpoints, templates, etc.

[0059] Hereinafter, general aspects of implementation of the systems and methods of the invention will be described.

[0060] The system of the invention or portions of the system of the invention may be in the form of a “processing machine,” such as a general purpose computer, for example. As used herein, the term “processing machine” is to be understood to include at least one processor that uses at least one memory. The at least one memory stores a set of instructions. The instructions may be either permanently or temporarily stored in the memory or memories of the processing machine. The processor executes the instructions that are stored in the memory or memories in order to process data. The set of instructions may include various instructions that perform a particular task or tasks, such as those tasks described above. Such a set of instructions for performing a particular task may be characterized as a program, software program, or simply software.

[0061] In one embodiment, the processing machine may be a specialized processor.

[0062] As noted above, the processing machine executes the instructions that are stored in the memory or memories to process data. This processing of data may be in response to commands by a user or users of the processing machine, in response to previous processing, in response to a request by another processing machine and/or any other input, for example.

[0063] As noted above, the processing machine used to implement the invention may be a general purpose computer. However, the processing machine described above may also utilize any of a wide variety of other technologies including a special purpose computer, a computer system including, for example, a microcomputer, mini-computer or mainframe, a programmed microprocessor, a micro-controller, a peripheral integrated circuit element, a CSIC (Customer Specific Integrated Circuit) or ASIC (Application Specific Integrated Circuit) or other integrated circuit, a logic circuit, a digital signal processor, a programmable logic device such as a FPGA, PLD, PLA or PAL, or any other device or arrangement of devices that is capable of implementing the steps of the processes of the invention.

[0064] The processing machine used to implement the invention may utilize a suitable operating system. Thus, embodiments of the invention may include a processing machine running the iOS operating system, the OS X operating system, the Android operating system, the Microsoft Windows™ operating systems, the Unix operating system, the Linux operating system, the Xenix operating system, the IBM AIX™ operating system, the Hewlett-Packard UX™ operating system, the Novell Netware™ operating system, the Sun Microsystems Solaris™ operating system, the OS/2™ operating system, the BeOS™ operating system, the Macintosh operating system, the Apache operating system, an OpenStep™ operating system or another operating system or platform.

[0065] It is appreciated that in order to practice the method of the invention as described above, it is not necessary that the processors and/or the memories of the processing machine be physically located in the same geographical place. That is, each of the processors and the memories used by the processing machine may be located in geographically distinct

locations and connected so as to communicate in any suitable manner. Additionally, it is appreciated that each of the processor and/or the memory may be composed of different physical pieces of equipment. Accordingly, it is not necessary that the processor be one single piece of equipment in one location and that the memory be another single piece of equipment in another location. That is, it is contemplated that the processor may be two pieces of equipment in two different physical locations. The two distinct pieces of equipment may be connected in any suitable manner. Additionally, the memory may include two or more portions of memory in two or more physical locations.

[0066] To explain further, processing, as described above, is performed by various components and various memories. However, it is appreciated that the processing performed by two distinct components as described above may, in accordance with a further embodiment of the invention, be performed by a single component. Further, the processing performed by one distinct component as described above may be performed by two distinct components. In a similar manner, the memory storage performed by two distinct memory portions as described above may, in accordance with a further embodiment of the invention, be performed by a single memory portion. Further, the memory storage performed by one distinct memory portion as described above may be performed by two memory portions.

[0067] Further, various technologies may be used to provide communication between the various processors and/or memories, as well as to allow the processors and/or the memories of the invention to communicate with any other entity; i.e., so as to obtain further instructions or to access and use remote memory stores, for example. Such technologies used to provide such communication might include a network, the Internet, Intranet,

Extranet, LAN, an Ethernet, wireless communication via cell tower or satellite, or any client server system that provides communication, for example. Such communications technologies may use any suitable protocol such as TCP/IP, UDP, or OSI, for example.

[0068] As described above, a set of instructions may be used in the processing of the invention. The set of instructions may be in the form of a program or software. The software may be in the form of system software or application software, for example. The software might also be in the form of a collection of separate programs, a program module within a larger program, or a portion of a program module, for example. The software used might also include modular programming in the form of object oriented programming. The software tells the processing machine what to do with the data being processed.

[0069] Further, it is appreciated that the instructions or set of instructions used in the implementation and operation of the invention may be in a suitable form such that the processing machine may read the instructions. For example, the instructions that form a program may be in the form of a suitable programming language, which is converted to machine language or object code to allow the processor or processors to read the instructions. That is, written lines of programming code or source code, in a particular programming language, are converted to machine language using a compiler, assembler or interpreter. The machine language is binary coded machine instructions that are specific to a particular type of processing machine, i.e., to a particular type of computer, for example. The computer understands the machine language.

[0070] Any suitable programming language may be used in accordance with the various embodiments of the invention. Illustratively, the programming language used may include assembly language, Ada, APL, Basic, C, C++, COBOL, dBase, Forth, Fortran, Java, Modula-2, Pascal, Prolog, REXX, Visual Basic, and/or JavaScript, for example. Further, it is not necessary that a single type of instruction or single programming language be utilized in conjunction with the operation of the system and method of the invention. Rather, any number of different programming languages may be utilized as is necessary and/or desirable.

[0071] Also, the instructions and/or data used in the practice of the invention may utilize any compression or encryption technique or algorithm, as may be desired. An encryption module might be used to encrypt data. Further, files or other data may be decrypted using a suitable decryption module, for example.

[0072] As described above, the invention may illustratively be embodied in the form of a processing machine, including a computer or computer system, for example, that includes at least one memory. It is to be appreciated that the set of instructions, i.e., the software for example, that enables the computer operating system to perform the operations described above may be contained on any of a wide variety of media or medium, as desired. Further, the data that is processed by the set of instructions might also be contained on any of a wide variety of media or medium. That is, the particular medium, i.e., the memory in the processing machine, utilized to hold the set of instructions and/or the data used in the invention may take on any of a variety of physical forms or transmissions, for example. Illustratively, the medium may be in the form of paper, paper transparencies, a compact disk, a DVD, an integrated circuit, a hard disk, a floppy disk, an

optical disk, a magnetic tape, a RAM, a ROM, a PROM, an EPROM, a wire, a cable, a fiber, a communications channel, a satellite transmission, a memory card, a SIM card, or other remote transmission, as well as any other medium or source of data that may be read by the processors of the invention.

[0073] Further, the memory or memories used in the processing machine that implements the invention may be in any of a wide variety of forms to allow the memory to hold instructions, data, or other information, as is desired. Thus, the memory might be in the form of a database to hold data. The database might use any desired arrangement of files such as a flat file arrangement or a relational database arrangement, for example.

[0074] In the system and method of the invention, a variety of “user interfaces” may be utilized to allow a user to interface with the processing machine or machines that are used to implement the invention. As used herein, a user interface includes any hardware, software, or combination of hardware and software used by the processing machine that allows a user to interact with the processing machine. A user interface may be in the form of a dialogue screen for example. A user interface may also include any of a mouse, touch screen, keyboard, keypad, voice reader, voice recognizer, dialogue screen, menu box, list, checkbox, toggle switch, a pushbutton or any other device that allows a user to receive information regarding the operation of the processing machine as it processes a set of instructions and/or provides the processing machine with information. Accordingly, the user interface is any device that provides communication between a user and a processing machine. The information provided by the user to the processing machine through the user interface may be in the form of a command, a selection of data, or some other input, for example.

[0075] As discussed above, a user interface is utilized by the processing machine that performs a set of instructions such that the processing machine processes data for a user. The user interface is typically used by the processing machine for interacting with a user either to convey information or receive information from the user. However, it should be appreciated that in accordance with some embodiments of the system and method of the invention, it is not necessary that a human user actually interact with a user interface used by the processing machine of the invention. Rather, it is also contemplated that the user interface of the invention might interact, i.e., convey and receive information, with another processing machine, rather than a human user. Accordingly, the other processing machine might be characterized as a user. Further, it is contemplated that a user interface utilized in the system and method of the invention may interact partially with another processing machine or processing machines, while also interacting partially with a human user.

[0076] It will be readily understood by those persons skilled in the art that the present invention is susceptible to broad utility and application. Many embodiments and adaptations of the present invention other than those herein described, as well as many variations, modifications and equivalent arrangements, will be apparent from or reasonably suggested by the present invention and foregoing description thereof, without departing from the substance or scope of the invention.

[0077] Accordingly, while the present invention has been described here in detail in relation to its exemplary embodiments, it is to be understood that this disclosure is only illustrative and exemplary of the present invention and is made to provide an enabling disclosure of the invention. Accordingly, the foregoing disclosure is not intended to be construed or to limit the

present invention or otherwise to exclude any other such embodiments, adaptations, variations, modifications or equivalent arrangements.

CLAIMS

What is claimed is:

1. A method for using an on-demand development environment, comprising:
in an information processing apparatus comprising at least one computer processor:
receiving, from a user interface, a request for an on-demand development environment for an application binary;
verifying that infrastructure for the on-demand development environment is not provisioned;
identifying a build template for the on-demand development environment;
selecting an endpoint for the on-demand development environment;
initiating the on-demand development environment on the selected endpoint using the build template; and
deploying the application binary to the on-demand development environment.
2. The method of claim 1, wherein the on-demand development environment is a container-based development environment.
3. The method of claim 1, further comprising determining whether a build template exists.
4. The method of claim 1, wherein the request includes at least one of a template, a source, a run command, and a port.

5. The method of claim 1, wherein the endpoint is selected based on at least one functional requirement.

6. The method of claim 5, wherein the function requirements comprise an identification one or more external services required for the application binary or a template received by the user interface.

7. The method of claim 1, wherein the endpoint is selected based on at least one non-functional requirement.

8. The method of claim 7, wherein the non-functional requirement includes at least one of a size of the on-demand development environment, an available capacity for the endpoint, a performance of the endpoint, a data sensitivity of the application binary, and a cost of using the endpoint.

9. The method of claim 1, wherein the build template comprises at least one of an identification of an operating system, an application infrastructure, an on-demand agent, a port, and debugging tool.

10. The method of claim 1, wherein the build template comprises a Docker image.

11. The method of claim 10, wherein the Docker image is retrieved from an image repository.

12. The method of claim 1, wherein the build template is selected based on a build template used in a prior request for an on-demand development environment.

13. The method of claim 1, wherein the endpoint is one of Marathon Mesos, Cloud Foundry, AWS, and Kubernetes.

14. The method of claim 1, further comprising:
deactivating the on-demand development environment.

15. The method of claim 13, further comprising:
saving a snapshot of state of the on-demand development environment.

16. The method of claim 1, further comprising:
collecting performance metrics about the on-demand development environment, wherein the performance metrics include at least one of a time for the on-demand development environment to go live, a performance of the on-demand development environment, and an available capacity of the on-demand development environment.

17. An on-demand development system, comprising:
a plurality of clients;
a server comprising at least one computer processor executing an on-demand application;
a template repository comprising a plurality of build templates; and
a plurality of endpoints, each end point executing an on-demand agent;

wherein:

the on-demand application receives a request for an on-demand development environment for an application binary from one of the clients;

the on-demand application verifies that infrastructure for the on-demand development environment is not provisioned;

the on-demand application identifies a build template for the on-demand development environment;

the on-demand application retrieves the build template from the template repository;

the on-demand application selects one of the plurality of endpoints for the on-demand development environment;

the on-demand application initiates the on-demand development environment on the endpoint using the build template;

the on-demand application deploys the application binary to the on-demand development environment; and

the agent for the selected endpoint unpacks the application binary to the on-demand development environment;

wherein the build template comprises a docker image.

18. The system of claim 17, wherein the on-demand development environment is a container-based development environment.

19. The system of claim 17, wherein the endpoint is selected based on at least one functional requirement including at least one of an identification one or more external services required for the application binary or a template received by the user interface.

20. The system of claim 17, wherein the endpoint is selected based on at least one non-functional requirement including at least one of a size of the on-demand development environment, an available capacity for the endpoint, a performance of the endpoint, a data sensitivity of the application binary, and a cost of using the endpoint.

1/2

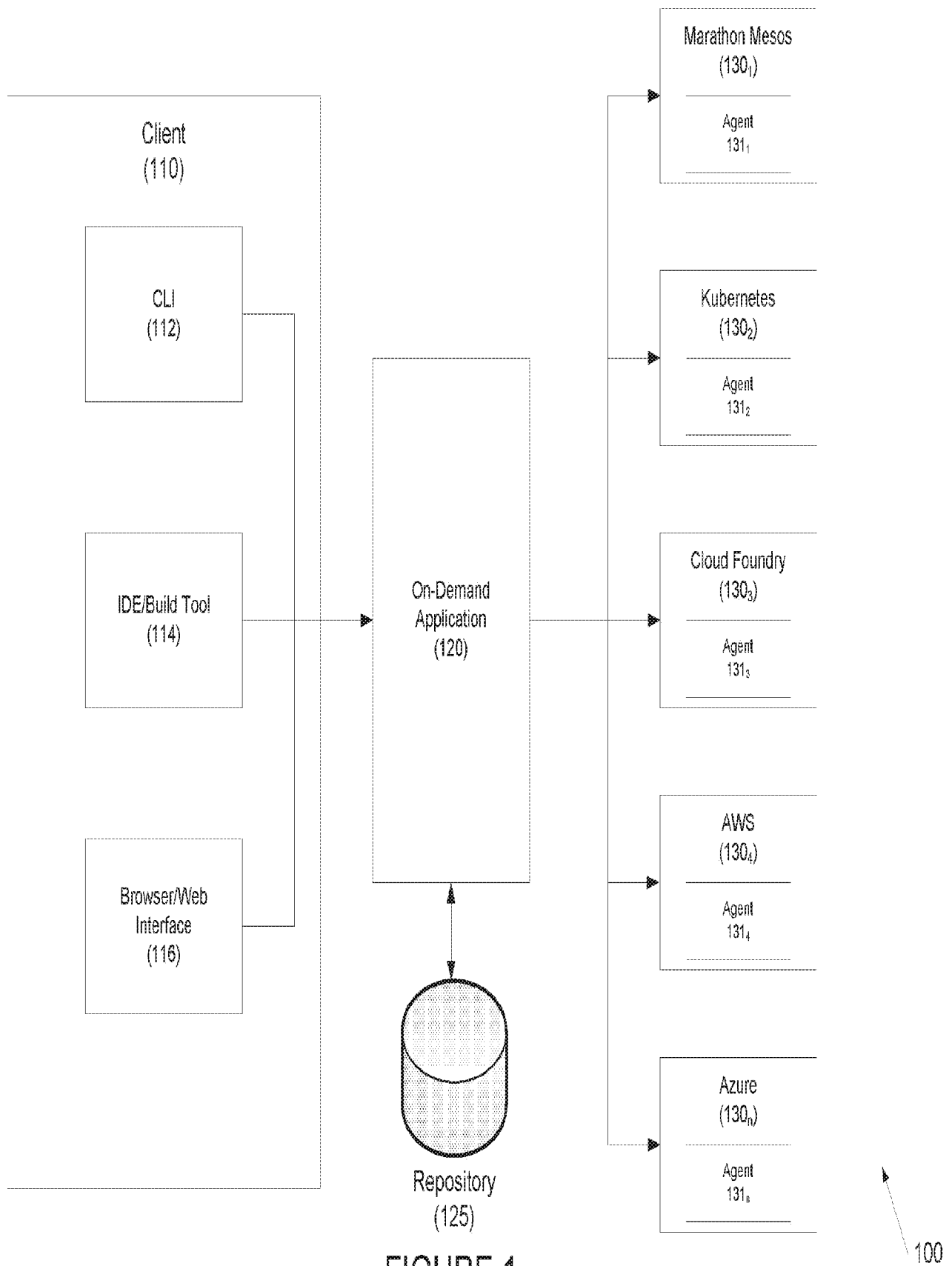


FIGURE 1

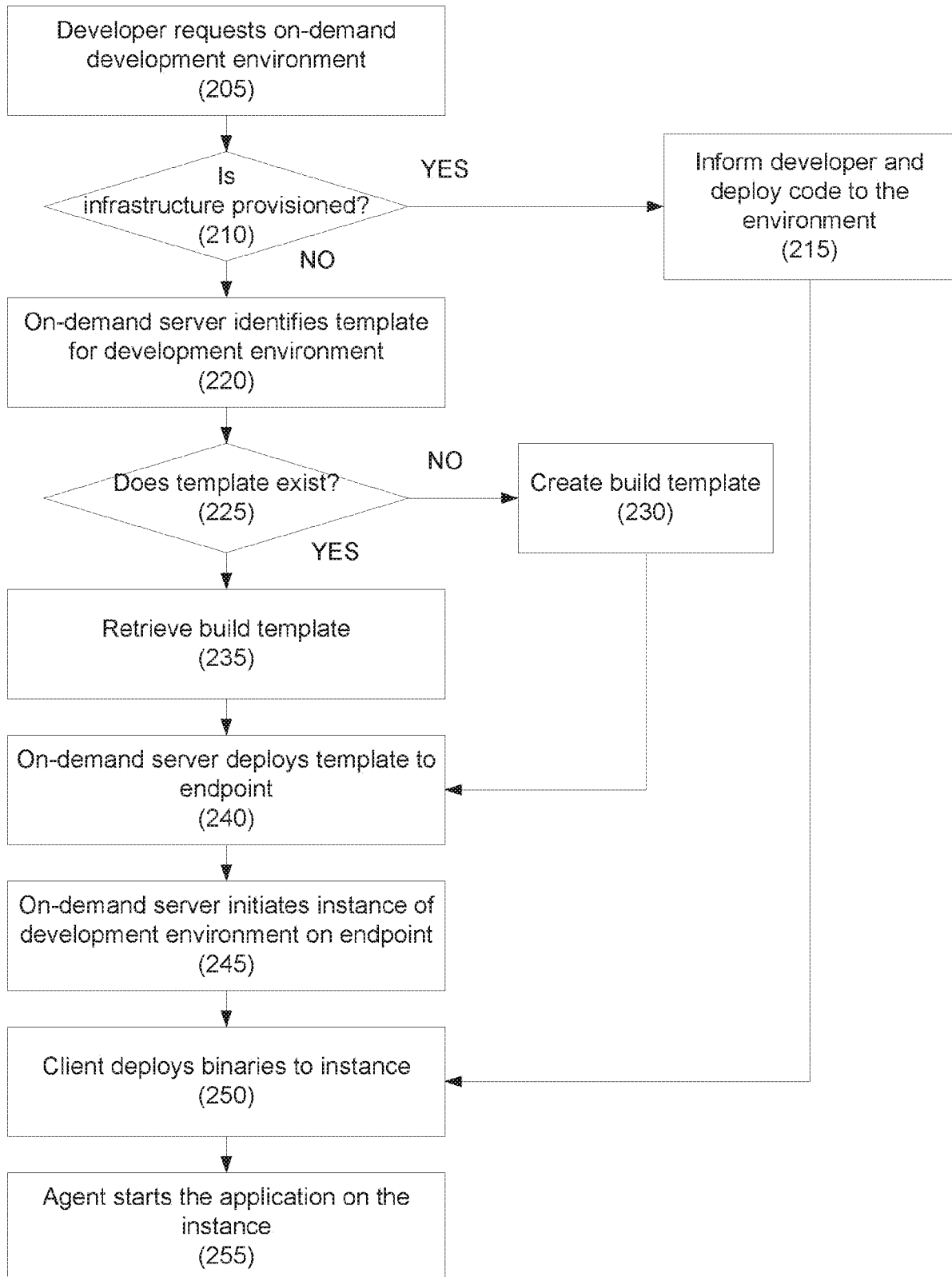


FIGURE 2

PATENT COOPERATION TREATY

PCT

INTERNATIONAL SEARCH REPORT

(PCT Article 18 and Rules 43 and 44)

Applicant's or agent's file reference 052227-500077	FOR FURTHER ACTION see Form PCT/ISA/220 as well as, where applicable, item 5 below.	
International application No. PCT/US2019/047482	International filing date (<i>day/month/year</i>) 21 August 2019 (21.08.2019)	(Earliest) Priority Date (<i>day/month/year</i>) 21 August 2018 (21.08.2018)
Applicant JPMORGAN CHASE BANK , N. A.		

This international search report has been prepared by this International Searching Authority and is transmitted to the applicant according to Article 18. A copy is being transmitted to the International Bureau.

This international search report consists of a total of 2 sheets.

☐ It is also accompanied by a copy of each prior art document cited in this report.

1. **Basis of the report**

a. With regard to the **language**, the international search was carried out on the basis of:

☒ the international application in the language in which it was filed.

☐ a translation of the international application into _____ which is the language of a translation furnished for the purposes of international search (Rules 12.3(a) and 23.1(b)).

b. ☐ This international search report has been established taking into account the **rectification of an obvious mistake** authorized by or notified to this Authority under Rule 91 (Rule 43.6bis(a)).

c. ☐ With regard to any **nucleotide and/or amino acid sequence** disclosed in the international application, see Box No. I.

2. ☐ **Certain claims were found unsearchable** (see Box No. II).

3. ☐ **Unity of invention is lacking** (see Box No. III).

4. With regard to the **title**,

☒ the text is approved as submitted by the applicant.

☐ the text has been established by this Authority to read as follows:

5. With regard to the **abstract**,

☒ the text is approved as submitted by the applicant.

☐ the text has been established, according to Rule 38.2, by this Authority as it appears in Box No. IV. The applicant may, within one month from the date of mailing of this international search report, submit comments to this Authority.

6. With regard to the **drawings**,

a. the figure of the **drawings** to be published with the abstract is Figure No. 1

☐ as suggested by the applicant.

☒ as selected by this Authority, because the applicant failed to suggest a figure.

☐ as selected by this Authority, because this figure better characterizes the invention.

b. ☐ none of the figures is to be published with the abstract.

INTERNATIONAL SEARCH REPORT

International application No.

PCT/US2019/047482

A. CLASSIFICATION OF SUBJECT MATTER

IPC: G06F 11/36, 8/76, 9/455

CPC: G06F 11/3688, 11/3664, 11/3684, 11/362, 11/3672

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

US-PGPUB; USPAT; USOCR;

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 2016/0179499 A1 (REEVES ET AL.) 23 June 2016 (23.06.2016) , entire document.	1-20
Y	US 9,542,219 B1 (BRYANT ET AL.) 10 January 2017 (10.01.2017) , entire document.	1-20
Y	US 2017/0277524 A1 (BHAT ET AL.) 28 September 2017 (28.09.2017) , entire document.	1-20

☐ Further documents are listed in the continuation of Box C.☐ See patent family annex.

* Special categories of cited documents: "A" document defining the general state of the art which is not considered to be of particular relevance "E" earlier application or patent but published on or after the international filing date "L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified) "O" document referring to an oral disclosure, use, exhibition or other means "P" document published prior to the international filing date but later than the priority date claimed	"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention "X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone "Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art "&" document member of the same patent family
---	---

Date of the actual completion of the international search 18 September 2019 (18.09.2019)	Date of mailing of the international search report 01 OCT 2019
Name and mailing address of the ISA/US COMMISSIONER FOR PATENTS MAIL STOP PCT, ATTN: ISA/US P.O. BOX 1450 ALEXANDRIA, VA 22313-1450, UNITED STATES OF AMERICA Facsimile No. (571)273-8300	Authorized officer HARRY KIM Telephone No. 571-272-4300