

"Systems methods, and computer program products for managing canary deployments"

Technical Field

[0001] The present disclosure relates to methods, systems and computer program products for managing canary deployments. Some embodiments relate to pushing, tracking and maintaining canary deployments in a live production environment.

Background

[0002] In software testing, a canary deployment, also known as canary testing, is a method of software testing that exposes a new feature or a change to an existing feature of an application to a sub-segment of the application's users. The goal of canary testing is to test new functionality or features on a subset of actual users before releasing to the entire user base. A canary deployment involves selecting a subset of a set of production computing devices, such as application servers, pushing a change to the code running on that subset of production computing devices, funnelling a selection of users to the subset of production computing devices and monitoring the performance of the change. Some methods of canary testing in live deployment environments may result in ineffective and/or less useful testing results.

[0003] It is desired to address or ameliorate some of the disadvantages associated with such prior methods of canary testing and systems, or at least to provide a useful alternative thereto.

[0004] Any discussion of documents, acts, materials, devices, articles or the like which has been included in the present specification is not to be taken as an admission that any or all of these matters form part of the prior art base or were common general knowledge in the field relevant to the present disclosure as it existed before the priority date of each of the appended claims.

Summary

[0005] Some embodiments are directed to a computer implemented method comprising: determining a first code portion; determining one or more second code portions; determining one or more code portion combinations by combining the first code portion with each of the second code portions; responsive to determining that a deployment code portion is being pushed to a plurality of computing devices, determining the deployment code portion; determining, based on the deployment code portion and the one or more code portion combinations, a candidate code portion that is indicative of the deployment code portion; causing the candidate code portion to be pushed to a sub-set of the plurality of computing devices.

[0006] In some embodiments, the method further comprises: determining two or more second code portions; and determining two or more code portion combinations.

[0007] In some embodiments, determining the first code portion comprises receiving, from a code development computing device, code indicative of the first code portion.

[0008] In some embodiments, determining the one or more second code portions comprises: determining one or more pull requests; and determining one or more code portions associated with each of the one or more pull requests.

[0009] In some embodiments, the pull request is indicative of a request to merge a branch of a code base associated with one of the second code portions with a main branch of the code base.

[0010] In some embodiments, the pull request comprises a canary identifier, and wherein the canary identifier indicates that the first code portion is permitted to be combined with the second code portion associated with the pull request.

[0011] In some embodiments, determining one or more code pull requests comprises: monitoring communications between one or more code development computing devices and one or more code distribution servers.

[0012] In some embodiments, determining the one or more code change combinations further comprises: prior to combining the first code portion with each of the second code portions, determining whether there is a conflict between the first code portion and one or more of the one or more second code portions.

[0013] In some embodiments, determining whether there is a conflict comprises determining whether the first code branch and one or more of the second code branches are associated with a same code section.

[0014] In some embodiments, if it is determined that there is a conflict, causing a warning to be sent.

[0015] In some embodiments, determining that the deployment code portion is being pushed to a plurality of computing devices comprises: receiving an indication from a code distribution server that the deployment code portion has been pushed or has been approved to be pushed.

[0016] In some embodiments, the indication is a web hook.

[0017] In some embodiments, the method further comprises: determining, from the indication, the second code portion or a universal resource indicator (URI) indicating the location of the second code portion.

[0018] In some embodiments, determining that the deployment code portion is being pushed to a plurality of computing devices comprises determining that a request to push the deployment code portion to the plurality of computing devices has been sent.

[0019] In some embodiments, determining that a request to push a second code change to a plurality of computing devices has been sent comprises, monitoring traffic of a communications network to determine if the request to push a second code change has been communicated.

[0020] In some embodiments, determining that a request to push the deployment code portion to a plurality of computing devices has been sent further comprises: subsequent to determining that the request to push the deployment code portion has been communicated, retrieving the deployment code portion from a computing device of a code development environment or a code distribution server.

[0021] In some embodiments, determining the candidate code portion comprises: comparing the deployment code portion or an indication of the deployment code portion to the one or more code portion combinations and determining which code portion combination is most similar to the deployment code portion.

[0022] In some embodiments, determining one or more second code portions comprises: receiving a notification that a second code portion has been deployed to a pre-production environment for preparation; and determining, based on the notification, a copy of the second code portion associated with the notification.

[0023] In some embodiments, determining the one or more code portion combinations comprises: subsequent to receiving the notification that that a second code portion has been deployed to a pre-production environment, and determining the copy of the second code portion associated with the notification; combining a copy of the first code portion with the copy of the second code portion associated with the notification; and submitting the combination of the first code portion with the copy of the second code portion to the pre-production environment.

[0024] In some embodiments, the first code portion, each of the one or more second code portions, each of the one or more code portion combinations, the deployment code

portion and the candidate code portion comprise one or more changes and/or additions to code of a codebase.

[0025] In some embodiments, the first code portion is indicative of code that is to be tested as part of a canary deployment.

[0026] Some embodiments are directed to a non-transitory machine-readable medium storing instructions which, when executed by one or more processors, cause a computing device to perform the method of the present disclosure.

[0027] Some embodiments are directed to one or more processors; and memory comprising computer executable instructions, which when executed by the one or more processors, cause the system to perform the method of the present disclosure.

[0028] Throughout this specification the word "comprise", or variations such as "comprises" or "comprising", will be understood to imply the inclusion of a stated element, integer or step, or group of elements, integers or steps, but not the exclusion of any other element, integer or step, or group of elements, integers or steps.

Brief Description of Drawings

[0029] Figure 1 is a process flow diagram showing a broad overview of a method for managing canary deployments, according to some embodiments;

[0030] Figure 2 is a block diagram of a system for managing canary deployments, according to some embodiments;

[0031] Figure 3 is a process flow diagram of a method for managing canary deployments, according to some embodiments; and

[0032] Figure 4 is a schematic diagram of a system for managing canary deployments, according to some embodiments.

Description of Embodiments

[0033] The present disclosure relates to methods, systems and computer program products for managing canary deployments. Some embodiments relate to pushing, tracking and maintaining canary deployments in a live production environment.

[0034] Canary deployments and canary testing are useful tools to develop stable, well performing, useable and/or likeable changes to software. For example, canary deployments can be used to test, in a real-life environment, using real-life users, a new product or feature of a digital service platform, without the risk of large scale problems with the platform, potentially stemming from the change. Problems that may stem from the change can range from user dissatisfaction with a minor change to a user interface, to wide spread crashes and down time of the digital services platform. This is achieved by pushing the change only to a limited subset of users or devices in a canary deployment. This involves generating a deployment code portion that includes the existing software and the new change, and pushing this out to the subset of users or devices in the form of a canary deployment. In addition, according to the present disclosure, canary deployments may be easily and quickly, for example, in a matter of minutes, be removed from deployment, without necessitating a remedying release to a fleet of servers.

[0035] However, in large and/or complex digital service platforms that have large teams of developers/software engineers working on the codebase, the development environment will have numerous code portions that are pending deployment to the production environment at any one time, and that may be released in numerous different orders. Accordingly, when code portions are pushed to a plurality of production computing devices, any previous code portions, including any previous canary deployments, may be overwritten and therefore there will no longer be an instance of the canary deployment present on any of the production machines. Further, canary deployments that exist in lively and complex development/production environments may not be long lived enough to produce test data of meaningful quality and/or quantity, functionally reducing the value of canary deployments and canary testing in general.

Similarly, canary deployments may also get scaled back as part of auto-scaling, for example, less traffic may lead to fewer instances, and the canary deployment may be an instance that is removed due to automation of deployment maintenance. Yet further, generating or otherwise preparing a canary deployment may take a non-trivial amount of time, and lead to unacceptable amounts of wait time for pushing not only canary deployments but other code portions as well. This may lead to slow response times to issues with the digital services platform.

[0036] Additionally or alternatively, a canary deployment may be being used to monitor platform or infrastructure-related change, such as changes that are solely to do with server side code. For example, the change being monitored may relate to validating that changing versions of dependencies, for example third party code libraries, does not result in any unexpected changes to functionality or performance, or any other unforeseen consequences. A single instance, in the form of a canary deployment, comprising the changes to be tested can be deployed to ensure the changes do not behave differently, to the rest of the fleet of servers.

[0037] Some embodiments of the present disclosure are directed to methods and systems for generating, deploying and maintaining canary deployments such that the canary deployment may be deployed quickly, regardless of the number of potential code portions that may be pending deployment to the production environment. Further, some embodiments of the present disclosure are directed to methods and systems for generating, deploying and/or maintaining canary deployments such that the canary deployment may be consistent or otherwise long lived, such that the canary deployment exists and is available for users to interact with for a sufficiently long period of time, and/or so as to generate a useable quantity and/or quality of testing data. The testing data being generated may be indicative of whether the code portions are functioning properly, and/or whether any changes to the digital services platform associated with the code portions are welcome changes by the users of the digital services platform. In other words, testing data may be indicative of such things as a change in efficiency, a change in platform stability, and/or a change in security, as induced by the code portions

associated with the canary deployment. In some embodiments, the testing data may be indicative of whether other parts and/or functionalities of the platform are working correctly. In other words, whether a change has had an unintentional impact on another aspects of the digital services platform. In some circumstances, the testing data may be indicative of whether any proposed changes to the digital services platform will or will not result in any significant crashes or system disruptions.

[0038] The present disclosure is directed to increased efficiency of code testing, development and deployment systems and methods. In particular, code portions that are to be deployed to a production environment need to go through testing and integration procedures to ensure that they are in a deployable state. Similarly, any canary deployment will need to go through the same processes, which takes a non-trivial amount of time. The code portions that are to be deployed will then be required to be held back from deployment until the canary deployment has been suitably prepared. The present disclosures are directed to developing canary deployments in parallel with a code development and testing, and deployment system to increase the efficiency of deploying canary deployments and other code portions.

[0039] The present disclosure is directed to improved canary deployments configured for increased life spans and maintainability or otherwise long lived canary deployments. Canary deployments that are already deployed to a development environment may be unintentionally, or unavoidably overwritten by a new code portion that is pushed out to a development environment. The present disclosures ensure that an already deployed canary deployment will be maintained by tracking new code portions to be deployed and pre-emptively determining canary deployments based on these new code portions. Thereby ensuring that when existing canary deployments are overwritten, the code that overwrites them comprises the same or substantially similar canary code.

[0040] In the present disclosure, the term code and/or code portion may refer to: an executable, a build, a portion of a line of code; a line of code in its entirety; several lines of code that are part of a function or an operation, such as a particular Object, or a

function of an Object and its associated contents; a code module comprising a plurality of lines of code operating in concert, a branch of a code base or code repository, or the entirety of a code base, for example. A code change, in the present disclosures may refer to any change to any code, including an alteration of existing code or the generation of new code, either as part of a new code module, or additional newly generated code that is added to an existing code module.

[0041] In the present disclosure, the term codebase may refer to code, that when executed by a computing device, provides the functionality of the digital services platform. The codebase may comprise all functioning code, and/or all code in current development, or otherwise all code that relates to the digital services platform, regardless of the life stage of the code in particular. Functioning code may be code that is currently deployed to a production environment and/or code that is capable of being deployed to the production environment. Code in current development may comprise any code that is being generated, changed, tested or any other process or step in the software development lifecycle.

[0042] **Figure 1** is a process flow diagram depicting a broad overview of a method 100 for managing canary deployments, according to some embodiments. For example, the method 100 may be executed by system 200, as discussed in further detail below with respect to Figure 2.

[0043] At **110** a canary code portion that is to be pushed out as part of a canary deployment is determined. This may be performed so that the canary code portion may be appropriately adapted such that it can be properly managed and tested. This may involve a coding team, such as a product development team, a product management team, a product testing team, a system security team and/or any other type of coding, software or management team that may be working on a digital service platform submitting a code portion for canary testing. Submitting a code portion, according to some embodiments, may comprise merging one or more code development branches into another code branch, such as a main branch or a testing branch. For example, merging a

side branch of a git repository with another side branch of the git repository. Examples of git repositories include GitHub, Bitbucket, GitLab, Amazon AWS CodeCommit, Microsoft Azure Repos. The repository, such as the git repository, may perform one or more conflict checks on the one or more side branches before merging it with the other side branch. In some embodiments, determining a canary code portion may be an automatic determination based on one or more predetermined rules for testing code. For example, an entity that develops and/or hosts a digital services platform may have requirements that certain code portions, such as code portions directed to changes to security, platform-related or infrastructure systems, and/or new product offerings, for example, are required to undergo canary testing before the code portion is permitted to be fully deployed to all the plurality of production computing devices.

[0044] In some embodiments, the canary code portion may have already been pushed out in a canary deployment, and may be actively running. In this circumstance, determining a canary code portion may comprise determining, by a canary deployment system, that one or more canary deployments have been previously deployed by the canary deployment system. In some embodiments, determining that a canary deployment has previously been deployed may comprise querying a stored set of canary code and/or canary deployments, wherein each canary code and/or canary deployment is indicative of a canary deployment that should be or already is deployed. Accordingly, in some embodiments, the stored set of canary code and/or canary deployments may be indicative of a current state or future state that should be maintained or achieved, respectively.

[0045] At **120** a set of possible deployment code portions that are pending release to the production environment is determined, for example by querying a repository of possible deployment code portions that are currently or have recently undergone deployment testing and integration processes and are therefore soon to be deployed to the production environment. This may be performed to anticipate any changes or additions that may be made to the code as deployed on the plurality of production computing devices to ensure that the canary code portion may be quickly deployed

regardless of the number of possible deployment code portions that are pending deployment.

[0046] Determining the set of possible deployment code portions that are pending release may comprise addressing a deployment queue that is indicative of a set of code portions that are pending release to the production environment of the digital services platform. The deployment queue may be determined by a code deployment system and/or process that is configured to generate code portions that are suitable for production deployment from pre-production code portions. In some embodiments, determining the set of possible code portions may comprise querying access logs for code portions from a code repository of the digital services platform.

[0047] At **130**, a set of canary code combinations is generated based on the canary code portion and the set of possible deployment code portions. This may be performed so that any possible deployment code portions that are to be pushed out to the plurality of production computing devices also comprise the canary code portion, which may allow the possible deployment code portions and the canary code portion to be quickly deployed. In some embodiments, generating a set of canary code combinations comprises combining the canary code portion with each of the possible deployment code portions. This may comprise adding the code that is indicative of the canary code portion to the code that is indicative of each of the potential code portions.

[0048] At **140** it is determined that one of the possible deployment code portions is going to be pushed out. In some embodiments, this may comprise determining that a request to push a code portion to the plurality of production computing devices has been sent and/or received. In some embodiments, determining that one of the possible deployment changes is going to be pushed out comprises determining that a request to push a code portion to the plurality of production computing devices has been sent and/or received. Determining that a request to push a code portion to the plurality of production computing devices has been sent and/or received may be performed by a canary system listening for events or otherwise observing communications between a

developer computing device and/or a code production system/process to a code distribution server. In some embodiments, at 140, the code portion that is to be pushed out is determined. The code portion that is to be pushed out may be determined based on the contents of the request to push the code change. In some embodiments, the contents of the request comprises the code portion that is to be pushed and/or a pointer to a location where the code portion that is to be pushed may be located, such as on a code development server or stored in a code development data store.

[0049] At **150**, a canary code combination that is indicative of the code portion that is to be pushed out is determined from the set of canary code combinations. This may comprise using the code portion that was determined from the request in step 140 above. The code portion that is to be pushed out is, at some point, caused to be deployed to the production environment. Responsive to the code portion that is to be pushed out being deployed, the determined canary code combination is deployed to one or more canary deployment computing devices. In this way, the canary deployment chases the code portion functionally immediately after the code portion is deployed. This leads to low lag times between code being marked to be deployed, and actually being deployed, by way of not having to wait for the canary deployment to be generate at the time of deployment.

[0050] In an exemplary embodiment of the present disclosure and specifically method 100, a first change or addition to the code of a digital services platform may be designated for eventual deployment to a production environment. Designating the first change or addition to the code may comprise the first change or addition to the code being submitted to a code pre-production pipeline. The code pre-production pipeline may comprise a plurality of steps and/or processes intended to test and/or prepare the first change or addition to the code for eventual deployment to servers of the digital services platform.

[0051] In some embodiments, upon the first change or addition to the code being submitted to the pre-production pipeline, a canary system may be alerted, such as by a canary tag or other canary identifier appended and/or otherwise associated with the

submission to the pre-production pipeline, and caused to perform a canary preparation process. Upon being alerted, the canary system will generate, copy, retrieve or otherwise receive a copy of the first change or addition to the code, for canary deployment purposes.

[0052] The canary system may, in some embodiments, append a portion of canary code to the first change or alteration to the code to determine a first canary deployment candidate. The canary system may then submit the first canary deployment candidate to the pre-production pipeline, such that the first canary deployment candidate is processed and prepared for deployment to the production environment in parallel with the first change or alteration to the code, thereby completing the preproduction process at substantially the same time. The first canary deployment candidate may comprise an identifier, that links the first canary deployment candidate with the first change or alteration to the code.

[0053] In some embodiments, the pre-production pipeline may receive a submission of a second change or alteration to the code of the digital services platform. The canary system may, upon receiving a notification, such as via a canary identifier, that a canary deployment candidate should be created, perform the same process that was performed for the first change or alteration to the code, to prepare a second canary deployment candidate. The second canary deployment candidate will then be caused to be stored with the first canary deployment candidate, in anticipation of potential deployment to the production environment.

[0054] The canary system may, according to some embodiments, be configured to perform the canary preparation process described above for all submissions to the pre-production pipeline. In this way, the canary system ensures that any change or alteration to the code submitted to the pre-production pipeline has a corresponding canary deployment candidate, that is tested and prepared in the same way as the associated change or alteration to the code, awaiting immediate deployment.

[0055] According to some embodiments, upon one of the changes or alterations to the code that has been submitted to the pre-production pipeline and passed the pre-production process being selected to be deployed to the production environment, a notification may be sent to a code deployment system. The notification may comprise an identifier that links the change or alteration to the code that is being deployed to its corresponding canary deployment candidate. Accordingly, the corresponding canary deployment candidate will chase the change or alteration to the code out to the production environment, to be deployed on a subset of the servers of the digital services platform.

[0056] In this way, the present disclosure enables the rapid deployment of canary deployments, rather than having to wait for a canary deployment to be built, and subsequently tested and prepared for deployment by the pre-production pipeline.

[0057] **Figure 2** is a block diagram of a system 200 for managing canary deployments, according to some embodiments. The system 200 comprises a canary system 210, network 240, code distribution server 250, database 260, production environment 270, and/or development environment 280. The production environment 270 may comprise production servers 272, network 274 and/or client device 276. The development environment 280 may comprise computing device(s) 282, network 284 and/or development servers 286.

[0058] The canary system 210 may comprise one or more computing devices and/or server devices, such as one or more servers, databases, and/or processing devices in communication over a network. The computing devices may host one or more application programs, libraries, APIs or other software elements. The canary system 210 comprises one or more processor(s) 210, network interface 214, and/or memory 216. The canary system 210 may be configured to receive, generate, configure, prepare, build or otherwise determine code portions, executables or builds that are to be pushed out as part of a canary deployment. The canary system 210 may be configured to receive code portions from database 260 and/or development environment 280, via network 240, for

determining canary deployments. In some embodiments, canary system 210 may receive and/or monitor for requests to push code portions to production environment 270 and/or monitor for when code portions are being or have been pushed to production environment 270. The canary system 210 may be configured, in some embodiments, to communicate canary deployments to code distribution server 250 for distribution to production environment 270. In some embodiments, canary system 210 may push canary deployments directly to production environment 270.

[0059] The processor(s) 212 may comprise one or more microprocessors, central processing units (CPUs), application specific instruction set processors (ASIPs), application specific integrated circuits (ASICs) or other processors capable of reading and executing instruction code.

[0060] The network interface 214 facilitates communications between canary system 210 and other components of system 200, such as code distribution server 250, database 260, production servers 272, and/or development environment 280, via network 240. The network interface 214 may comprise a combination of network interface hardware and network interface software suitable for establishing, maintaining and facilitating communication over a relevant communication channel.

[0061] The memory 216 may comprise one or more volatile or non-volatile memory types. For example, memory 216 may comprise one or more of random access memory (RAM), read-only memory (ROM), electrically erasable programmable read-only memory (EEPROM) or flash memory. The memory 216 is configured to store program code accessible by the processor(s) 212. The program code comprises executable program code modules. In other words, memory 216 is configured to store executable code modules configured to be executable by the processor(s) 212. The executable code modules, when executed by the processor(s) 212 cause the canary system 210 to perform certain functionality, as described in more detail below. For example, memory 216 may comprise a canary module 218, codebase module 220, , canary generation module 224, trigger module 226 and/or canary deployment module 228. A code module may be a

collection of computer executable or computer intelligible code, that when executed by a processor, causes the processor to perform processes, actions or functionalities, such as the processes, actions or functionalities described below.

[0062] The canary module 218 is configured to receive, generate, configure, prepare or otherwise determine code portions that are to be pushed out as a canary deployment, also referred to as canary code portions or first code portions. In some embodiments, canary module 218 may receive, from development environment 280 and/or database 260, code portions that are to be pushed out in a canary deployment.

[0063] The codebase module 220 is configured to track or otherwise determine a set of possible deployment code portions that have the potential to be pushed out to the production environment 270. The set of possible deployment code portions may comprise a list of all, most or some of the code and/or code changes that are awaiting deployment to the production environment 270. The set of possible deployment code portions may be a list of uniform resource identifiers (URIs) that are indicative of where the code portions are stored or where the code portions can otherwise be located, such as code distribution server 250, database 260 and/or development environment 280. In some embodiments, the codebase module 220 may be configured to monitor communications between computing device(s) 282 and development servers 286 and/or database 260 to determine requests for code portions that make up a larger codebase. For example, the development servers 286 may be a git repository, and the codebase module 220 may be configured to listen for an event, such as push and/or pull requests sent to the git repository, to track or otherwise determine what code is being accessed so that it be changed, added to and/or newly generated. In some embodiments, the codebase module 220 may be configured to query the code distribution server 250 to determine a schedule or list of planned or otherwise pending code portions to be pushed out to production environment 270.

[0064] The canary generation module 224 is configured to generate or otherwise determine one or more combinations of the canary code portion and each of the set of

possible deployment code portions. In some embodiments, the canary generation module 224 is configured to combine the canary code portion with each of the possible deployment code portions to determine a set of canary code combinations. Each of the set of canary code combinations may comprise the equivalent possible deployment code portions in addition to the canary code portion. In some embodiments, the canary generation module 224 is configured to treat the code associated with the canary code portion preferentially. In some embodiments, if there is a conflict between the canary code portion and a possible deployment code portions the system may generate and display a warning, such as on or via computing device(s) 282 of development environment 280.

[0065] The trigger module 226 is configured to determine that a trigger event has occurred. A trigger event may be a code portion is being, or as recently been pushed to the production environment 270 and/or that a request to push a code portion to the production environment 270 has been sent and/or received by one or more computing devices of the system 200. . In some embodiments, the trigger module 226 may be configured to monitor for activity indicative of a code portion being pushed to production environment 270. In some embodiment, the trigger module 226 may monitor network traffic of network 240 to determine whether a request to push a code portion has been sent and/or received. In some embodiments, the trigger module 226 may intercept all requests to push code portions, such as requests sent from development environment 280, and then forward the request on to code distribution server 250. In some embodiments, the trigger module 226 may be configured to monitor communications being received by code distribution server 250 ad/or sent from development environment 280. The trigger module 226, in some embodiments, is configured to determine the code portion that has been requested to be pushed to the production environment 270, otherwise referred to as the deployment code portion.

[0066] The canary deployment module 228 is configured to determine a canary deployment, also referred to as a candidate code portion. The candidate code portion is determined based on the canary code portion and the set of possible deployment code

portions. The canary deployment module 228 may determine the candidate code portion by determining which of the code portion combinations is indicative of the code portion that is to be pushed to the production environment 270. In some embodiments, the canary deployment module 228 is configured to prepare or otherwise ready the candidate code portion for deployment to the production environment 270. In some embodiments, the canary deployment module 228 is configured to determine, from production environment 270, one or more candidate computing devices that the canary deployment is to be pushed to. The canary deployment module 228, in some embodiments, may be configured to communicate canary deployments to the code distribution server 250, for pushing out to the production environment 270, for example by the push module 258.

[0067] The network 240 may include, for example, at least a portion of one or more networks having one or more nodes that transmit, receive, forward, generate, buffer, store, route, switch, process, or a combination thereof, etc. one or more messages, packets, signals, some combination thereof, or so forth. The network 240 may include, for example, one or more of: a wireless network, a wired network, an internet, an intranet, a public network, a packet-switched network, a circuit-switched network, an ad hoc network, an infrastructure network, a public-switched telephone network (PSTN), a cable network, a cellular network, a satellite network, a fibre-optic network, some combination thereof, or so forth.

[0068] The code distribution server 250 comprises one or more processor(s) 252, network interface 254 and/or memory 256. The code distribution server 250 is configured to push code portions to production environment 270. In some embodiments, code distribution server 250 may receive code portions that are to be pushed and prepare them for deployment. In some embodiments, code distribution server 250 is configured to receive requests to push code portions to the production environment 270, for example, from the canary system 210 and/or development environment 280. The code distribution server 250, in some embodiments, may be configured, in response to receiving a request to push code portions to the production environment 270, to retrieve

the code to be pushed from another system/device, such as the canary system 210, database 260 and/or development environment 280.

[0069] The processor(s) 252 may comprise one or more microprocessors, central processing units (CPUs), application specific instruction set processors (ASIPs), application specific integrated circuits (ASICs) or other processors capable of reading and executing instruction code.

[0070] The network interface 254 facilitates communications between code distribution server 250 and other components of the system 200, such as the canary system 210, database 260, hosting servers 270, and/or development environment 280, via network 240. The network interface 254 may comprise a combination of network interface hardware and network interface software suitable for establishing, maintaining and facilitating communication over a relevant communication channel.

[0071] The memory 256 may comprise one or more volatile or non-volatile memory types. For example, the memory 256 may comprise one or more of random access memory (RAM), read-only memory (ROM), electrically erasable programmable read-only memory (EEPROM) or flash memory. The memory 256 is configured to store program code accessible by the processor(s) 252. The program code comprises executable program code modules. In other words, the memory 256 is configured to store executable code modules configured to be executable by the processor(s) 252. The executable code modules, when executed by the processor(s) 252 cause the code distribution server 250 to perform certain functionality, as described in more detail below. For example, the memory 256 may comprise push module 258. A code module may be a collection of computer executable or computer intelligible code, that when executed by a processor, causes the processor to perform processes, actions or functionalities, such as the processes, actions or functionalities described below.

[0072] The push module 258 is configured to push or otherwise distribute code portions to the production environment 270, such as the production servers 272 and/or

client devices 276. The push module 258, in some embodiments, may receive code portions and prepare the code portions for deployment to the production environment 270. Preparing the code portions for deployment to the production environment 270 may comprise one or more of: compiling or building the code; combining disparate code; conducting testing such as unit tests, integration tests, end-to-end tests, functional tests, acceptance tests, performance tests, and/or any other types of relevant tests; creating one or more back-ups and/or roll back points; and/or removing unneeded code. In some embodiments, the push module 258 may be configured to use one or more URI, which may be communicated to the push module via the request to push, and retrieve one or more code portions from one or more code repository, such as the canary system 210, database 260 and/or development environment 280, for deployment to the production environment 270.

[0073] The database 260 may form part of or be local to the system 200, or may be remote from and accessible to the system 200, for example, via the network 240. The database 260 may be configured to store data associated with the system 200. The database 260 may be a centralised database. The database 260 may be a mutable data structure. The database 260 may be a shared data structure. The database 260 may be a data structure supported by database systems such as one or more of PostgreSQL, MongoDB, and/or ElasticSearch. The database 260 may be configured to store a current state of information or current values associated with various attributes (e.g., “current knowledge”). In some embodiments, system 200 may comprise more than one data store. For example, production environment 270 and/or development environment 280 may comprise one or more data stores (not shown).

[0074] The production environment 270 may be a collection of configured clusters, servers, databases, and/or other computing devices that collaborate to provide an environment that enables users to use or otherwise engage with a software application and/or digital services platform. The production environment 270 comprises production servers 272, network 274 and/or client device(s) 276.

[0075] The production servers 272 may comprise one or more computing devices configured to provide functionality for other programs or devices, (i.e. clients). The production servers 272 may comprise a client-server architecture. The production servers 272 may provide various functionalities, often called "services," such as sharing data or resources among one or more client device(s) 276. A single hosting server may serve multiple client device(s) 276. Production servers 272 may comprise one or more database servers, file servers, mail servers, print servers, web servers, game servers, and/or application servers, for example.

[0076] The production servers 272 may be configured to communicate or otherwise provide functionalities to client device(s) 276 via a request-response model. In a request-response model, client device(s) 276 send requests to the production servers 272, which performs some action and sends a response back to the client device(s) 276, typically with a result or acknowledgment. Each of production servers 272 may be a computing device designated as "server-class hardware", meaning that it is specialized for server actions. Each of development servers 272 may alternatively be a large computing cluster, composed of many relatively simple, replaceable server components.

[0077] The network 274 may include, for example, at least a portion of one or more networks having one or more nodes that transmit, receive, forward, generate, buffer, store, route, switch, process, or a combination thereof, etc. one or more messages, packets, signals, some combination thereof, or so forth. The network 274 may include, for example, one or more of: a wireless network, a wired network, an internet, an intranet, a public network, a packet-switched network, a circuit-switched network, an ad hoc network, an infrastructure network, a public-switched telephone network (PSTN), a cable network, a cellular network, a satellite network, a fibre-optic network, some combination thereof, or so forth.

[0078] The client device(s) 276 may comprise a mobile or handheld computing device such as a smartphone or tablet, a laptop, or a PC, and may, in some embodiments, comprise multiple computing devices. Client device(s) 276 may comprise one or more

pieces of application software (not shown), that enables the client device(s) 276 and/or a user to avail of the functionalities and/or services provided by the digital services platform as hosted by production servers 272. The client device(s) 276 may download the application software via an app store such as the Apple app store or the Google Play app store, or may download the application via a web browser, such as the Chrome web browser or the Safari web browser. In some embodiments, client device(s) 276 may interact with the digital service platform via a web-browser via a browser based application.

[0079] The development environment 280 may be a collection of configured clusters, servers, databases, and/or other computing devices that collaborate to provide an environment that allows software developers and coders to generate and develop code. The development environment 280 may comprise computing device(s) 282, network 284 and/or development servers 286. The development environment 280 may be configured to facilitate the generation, adaptation, testing and/or deployment of compute code that, once deployed and executed in the production environment, provide the functionality of a digital service platform.

[0080] The computing device(s) 282 may be one or more of a mobile or handheld computing device such as a smartphone or tablet, a laptop, or a PC, and may, in some embodiments, comprise multiple computing devices. The computing device(s) 282 may be configured to perform functions associated with code development. Computing device(s) 282 may be configured to request code portions from development servers 286. Computing device(s) 282 may be configured to receive code portions from development servers 286, such as in response to sending a request for code, so as to allow programmers to make changes or additions to the code, to develop and/or refine the functionalities enabled by the code portions, when the code portions are deployed to production environment 270. In some embodiments, computing device(s) 282 may be configured to communicate with canary system 210 and/or code distribution server 250. Computing device(s) 282 may communicate code changes, such as canary code changes and/or code push requests to canary system 210. In some embodiments, computing

device(s) 282 may communicate code changes and/or requests to push code to code distribution server 250.

[0081] The network 284 may include, for example, at least a portion of one or more networks having one or more nodes that transmit, receive, forward, generate, buffer, store, route, switch, process, or a combination thereof, etc. one or more messages, packets, signals, some combination thereof, or so forth. The network 284 may include, for example, one or more of: a wireless network, a wired network, an internet, an intranet, a public network, a packet-switched network, a circuit-switched network, an ad hoc network, an infrastructure network, a public-switched telephone network (PSTN), a cable network, a cellular network, a satellite network, a fibre-optic network, some combination thereof, or so forth. In some embodiments, networks 240, 274 and/or 284 may all be part of the same network. The networks 240, 274 and/or 284 may be separate networks.

[0082] The development servers 286 may comprise one or more computing devices configured to provide functionality for other programs or devices, (e.g. clients). The development servers 286 may comprise a client-server architecture. The development servers 286 may provide various functionalities, often called "services," such as sharing data or resources among the one or more computing device(s) 282. A single hosting server may serve multiple computing device(s) 282. A client process may run on the same device or may connect over network 284 to computing devices 282. Development servers 286 may comprise one or more database servers, file servers, mail servers, print servers, web servers, and/or application servers, for example.

[0083] The development servers 286 may be configured to communicate or otherwise provide functionalities to computing device(s) 282 via a request-response model. In a request-response model, computing device(s) 282 send requests to the development servers 286, which performs some action and sends a response back to the computing device(s) 282, typically with a result or acknowledgment. Each of development servers 286 may be a computing device designated as "server-class hardware", meaning that it is

specialized for server actions. Each of development servers 286 may alternatively be a large computing cluster, composed of many relatively simple, replaceable server components.

[0084] **Figure 3** is a process flow diagram of a method 300 for managing canary deployments, according to some embodiments. In some embodiments, system 200 may be configured to perform the method 300. Method 300 may correspond to broad overview method 100, and may describe the steps of method 100 in more detail.

[0085] At 310, canary system 210 determines a first code portion. The first code portion may be code change that is to be included in a canary deployment and may be also referred to as a candidate canary code portion, a canary code portion or simply canary code. The first code portion may be sent and/or received from development environment 280. In some embodiments, the first code portion may be sent and/or received from computing device(s) 282. The first code portion may be accessed and/or received from database 260. The first code portion may comprise one or more changes, alterations, adaptations, modifications, additions or variations to code that is part of the codebase. In some embodiments, canary system 210 receives a copy of the first code portion. The canary system 210 may, in some embodiments, receive a URI indicating the location of the first code portion. Responsive to receiving the URI, the canary system 210 may retrieve a copy of the first code portion using the received URI. In some embodiments, 310 and/or any of the processes of 310 may be performed by processor(s) 212, executing canary module 218. According to some embodiments, 310 may correspond to 110 of method 100.

[0086] At 315, the canary system 210 determines one or more second code portions. The one or more second code portions may be potential changes, alterations, adaptations, modifications, additions or variations to the code of the codebase. The one or more second code portions may be determined by the canary system 210 by listening for or otherwise monitoring communications associated with a second code portion being deployed to an environment. In some embodiments, the communications may be

associated with pushing or deploying the second code portions to the production environment 270. In some embodiments the communications may be associated with deploying a second code portion to a preproduction environment (not shown) or otherwise preparing the second code portions for deploying to the production environment 270. In some embodiments, the one or more second code portions may be determined by the canary system 210 by listening to events, such as communications with the code distribution server 250 indicative of requests to merge a particular branch of a code base with the main branch of a code base. The communications may comprise a canary tag or other canary identifier. The communications may be sent from the development environment 280 to the code distribution server 250. The canary identifier may indicate that the second code portion associated with the canary identifier is permitted and/or desired to be combined with the first code portion, as described in further detail below. In some embodiments, subsequent to determining that a communication has been sent, the canary system 210 may determine, from based on the communication, the second code portion associated with the particular communication.

[0087] In a further exemplary embodiment, the canary system 210 may monitor communications between the computing device(s) 282 and development servers 286, such as via network 284. In some embodiments, the development servers 286 may be one or more code repositories, such as git repositories, and the computing device(s) 282 may send push and/or pull requests to the one or more code repositories to download or upload portions of code from the code repositories. The canary system 210 may be configured to monitor the pull requests sent to the development servers 286, and log the one or more portions of code that have been deployed. In some embodiments, the code distribution server 250 may comprise a set of code portions that are to be pushed out to production environment 270. The canary system 210 may receive the set and/or an indication of the one or more second code portions from code distribution server 250, such as via network 240. In some embodiments, each pull and/or push request to the development servers 286 may have a canary identifier appended to otherwise added to the request, such that the code portions associated with each pull and/or push request can be tracked as a potential second code portion to potentially be included in the set of

second code portions. In some embodiments, the canary identifier may be indicative of the second code portion having an associated potential canary deployment, comprising the first code portion and the respective second code portion.

[0088] In some embodiments, at 315, the canary system 210 may determine two or more, or otherwise a plurality of second code portions. Each of the plurality of second code portions may comprise different code to that of any other second code portion of the plurality of second code portions. Each second code portion may be associated with different pull requests to merge a particular branch of a code base with the main branch of a code base. Each second code portion may comprise a unique canary identifier.

[0089] According to some embodiments, determining one or more second code portions comprises a second code portion being received by a pre-production pipeline (not shown) to go through a pre-production process. The pre-production pipeline and the pre-production process may be configured to test and prepare the second code portion for deployment to the production environment 270. The second code portion, when received by the pre-production pipeline may comprise a canary identifier. The canary identifier indicating that a canary deployment comprising the associated second code portion should be generated or otherwise created.

[0090] In some embodiments, the canary system 210 and/or the code distribution server 250 may comprise one or more monitoring rules indicative of when a second code portion should be determined for inclusion in a canary deployment. The monitoring rules may be associated with the presence or absence of a particular tag, marker or otherwise identifier with the second code portion. The monitoring rules may be associated with a system status, such as a status of the production environment 270, status of a pre-production environment (not shown), and/or any other applicable status. The monitoring rules may be associated with the type or nature of the code change associated with the second code portion. In some embodiments, monitoring rules may be associated with the anticipated changes that may be caused by the deployment of the second code portion.

[0091] In some embodiments, 315 and/or any of the processes of 315 may be performed by processor(s) 212, executing codebase module 220. According to some embodiments, 315 may form part of step 120 of method 100.

[0092] At 320, the canary system 210 is configured to determine, based on the first code portion and the one or more second code portions, one or more code portion combinations. Each of the one or more code portion combinations comprises a second code portion of the one or more second code portions and a copy of the first code portion. To determine the one or more code portion combinations, the canary system 210 may iteratively combine the first code portion with each of the second code change portions. In some embodiments, when the first code portion and a second code portion of the one or more second code portions comprise one or more changes associated with a same code section, the changes associated with the first code portion may be given priority. Giving priority to the changes associated with the first code portion may comprise selectively including the changes associated with the first code portion that conflict with the changes associated with the second code portion. In some embodiments, wherein the canary system 210 has determined two or more second code portions, or otherwise a plurality of code portions, the canary system 210 configures two or more code portion combinations. Each of the plurality of code portion combinations comprises a second code portion of the two or more second code portions and a copy of the first code portion.

[0093] In some embodiments, subsequently or contemporaneously to the second code portion being submitted to the pre-production pipeline, the canary system 210 may be notified that a code portion combination should be determined. The code portion combination is based on the first code portion and the second code portion that has been submitted to the pre-production pipeline. According to some embodiments, the canary identifier may cause the canary system 210 to be notified.

[0094] The canary system 210, according to some embodiments, may create the code portion combination and submit it to the pre-production pipeline for testing and

preparation in parallel with the associated second code portion. Upon completion of the pre-production process, the second code portion and the code portion combination may be caused to be stored in anticipation of deployment to the production environment 270.

[0095] In some embodiments, 325 and/or any of the processes of 335 may be performed by processor(s) 212 executing canary generation module 224. According to some embodiments, 325 may form part of 130 of method 100.

[0096] At 322, the canary system 210 determines that a deployment code portion is being pushed to the production environment 270. Determining that the deployment code portion is being pushed to the production environment 270 may comprise monitoring the behaviour of the code distribution server 250. Monitoring the behaviour of the code distribution server 250 may comprise monitoring for when the code distribution server 250, or the push module 258, causes the deployment code portion to be pushed to the production environment 270. In some embodiments, the canary system 210 may receive a notification via a web hook from code distribution system 250 that a particular code portion is being deployed to the production environment 270.

[0097] In some embodiments, determining that a code portion is being pushed to the production environment 270 may comprise determining that a request to push the deployment code portion to the development environment 270 has been sent. Determining that a request to push the deployment code portion to the development environment 270 has been sent may comprise receiving one or more requests from one or more component of system 200. In some embodiments, all, most or some of the requests to push deployment code portions to the development environment 270 may first be sent to canary system 210 to determine the deployment code portion. In some embodiments, the canary system 210 may monitor communications, such as communications via network 240 for communications that are indicative of a request for the deployment code portion to be pushed to production environment 270. Determining the deployment code portion may comprise receiving, by the canary system 210, the request to push the deployment code portion, wherein the request comprises a copy of

the deployment code portion. According to some embodiments, 325 may form part of 140 of method 100. . In some embodiments, 322 and/or any of the processes of 32, may be performed by processor(s) 212 executing trigger module 226.

[0098] In some embodiments, determining that a request to push the deployment code portion to the development environment 270 has been sent may comprise the canary system 210 listening, or otherwise monitoring network traffic, such as traffic being sent over network 240, to determine whether a request to push the deployment code portion has been sent, received or otherwise communicated. If the canary system 210 is configured to monitor network traffic, the canary system 210 may not be configured to receive push requests, as described above. In some embodiments, the canary system 210 may be configured to both receive push requests and monitor network traffic. The code distribution server 250 may, in some embodiments, subsequent to receiving a request to push the deployment code portion to the production environment, send the request, or another type of notification or indication to the canary system 210, indicating that the request has been received.

[0099] At 325, the canary system 210, subsequent to determining that a deployment code portion is being pushed to the production environment 270, determines the deployment code portion. In some embodiments, determining the deployment code portion comprises receiving the request to push the deployment code portion, the request comprising a URI, indicating where the deployment code portion is stored or otherwise located. Subsequent to determining the URI from the request, the canary system 210 may be configured to locate the deployment portion using the URI and thereby retrieve a copy of the deployment code portion. In some embodiments, the canary system 210 may receive a notification from code distribution server 250 indicative of the code deployment portion. For example, the notification may comprise an identifier that is indicative of the code portion combination associated with the deployment code portion that is being pushed to the production environment 270.

[0100] In some embodiments, 325 and/or any of the processes of 325, may be performed by processor(s) 212 executing trigger module 226. According to some embodiments, 325 may form part of 140 of method 100.

[0101] At 330, the canary system 210 determines a candidate code portion from the one or more code portion combinations that is indicative of the deployment code portion. Determining a candidate code portion may comprise comparing the deployment code portion to the one or more code portion combinations to determine the code portion combination the is most similar to the deployment code portion. In some embodiments, the web hook indicative of the deployment code portion comprises an identifier that is associated with the equivalent code portion combination.

[0102] In some embodiments, 330 and/or any of the processes of 330, may be performed by processor(s) 212 executing trigger module 226. According to some embodiments, 325 may form part of 150 of method 100.

[0103] At 335, the canary system 210 causes the candidate code portion to be pushed out to the production environment 270. In some embodiments, the canary system 210 is configured to determine at least one candidate computing device from the production environment 270. Determining the at least one candidate computing device may comprise determining a plurality of computing devices that the candidate code portion is to be pushed to, for example, production server(s) 272 and/or client device(s) 276. The at least one candidate computing device may be determined from the plurality of computing devices. In some embodiments, the candidate computing device may be selected at random from the plurality of computing devices.

[0104] In some embodiments, determining the candidate computing device comprises determining one or more candidate computing device attributes. Candidate computing device attributes may include, for example, the device's geographical location, computational capabilities, currently and/or previously deployed code and/or canary deployments, and/or any other suitable attribute. For example, the first code portion may

be associated with changes to one or more UK specific features of the digital services platform, accordingly, a computing device of the plurality of computing devices that serves UK users may be selected as the candidate computing device upon which the candidate code portion is deployed. While the above exemplary embodiments mention a single candidate computing device, it should be understood that a plurality of candidate computing devices may be selected from the plurality of computing devices.

[0105] In some embodiments, 335 and/or any of the processes of 335, may be performed by processor(s) 212 executing canary deployment module 228. According to some embodiments, 335 may form part of 150 of method 100.

[0106] Optionally at 340, the canary system 210 may determine or otherwise generate, deploy or implement a filter, configured to direct particular users to the candidate computing device and other users to the plurality of other computing devices, such as the production servers 272. Users may be directed by the filter to the candidate computing device based on one or more user attributes, such as geographical location.

[0107] In some embodiments, the canary system 210 causes the candidate code portion to be pushed out to the candidate computing devices by transferring a copy or copies of the candidate code portion to code distribution server 250, which may then push out the candidate code portion to the candidate computing device. In some embodiments, the candidate code portion may be pushed out by processor(s) 252 executing push module 258. According to some embodiments, 340 may form part of 150 of method 100.

[0108] In some embodiments, the canary system 210 may be part of code distribution server 250, such as a physical component housed in the same collection of computing devices that constitutes code distribution server 250. In some embodiments, the canary system 210 may be a software module running on the code distribution server 250. In some embodiments, the code distribution server 250 may be a part of the canary system 210, such as a physical component housed in the same collection of computing devices

that constitutes the canary system 210. In some embodiments, the code distribution server 250 may be a software module running on canary system 210.

[0109] In some embodiments, 340 or any of the processes of 340 may be performed by processor(s) 212 executing canary deployment module 228. According to some embodiments, 340 may form part of 150 of method 100.

[0110] **Figure 4** is a schematic diagram 400 depicting a system for managing canary deployments, according to some embodiments. Schematic diagram 400 depicts development environment 280, canary system 210, code distribution server 250 and production environment 270. As depicted in Figure 2, the production environment 270 comprises production servers 272 and client device(s) 276. The development environment 280 may comprise codebase 410.

[0111] The production servers 272 may comprise standard production servers 422 and the canary deployment server 425. The standard production servers 422 may comprise deployed code 427. In some embodiments, deployed code 427 may comprise the deployment code portion 432, that has been pushed out by the code distribution server 250. The deployed code 427, when executed by the production servers 272, or standard production servers 422, may provide the functionality of the digital services platform. The canary production server 425 may comprise deployed code 430. In some embodiments, the deployed code 430 may comprise the canary deployment 435 (candidate code portion) determined according to the method 300. The deployed code 430, when executed by the canary deployment server 425, may provide the functionality of the digital services platform, in accordance with the changes to functionalities, or changes to user experience, or any other type of changes, that are being tested via the canary deployment and as defined by the candidate code portion. While only two standard production servers 422 and one canary deployment server 425 are illustrated, it is to be understood that servers 272 may comprise any number of standard production servers 422 and canary deployment servers 425.

[0112] As depicted by Figure 4, code portions, such as the first code portion, the set of second code portions, and/or the deployment code portion are determined from development environment 280. For example, the first code portion may be determined from the development environment 280 by a testing team, or a software engineer that wishes to generate a canary deployment to test one or more changes to the digital services platform, as described above. The set of second code portions may be determined from requests for access and/or copies of the codebase 410, such as by a software engineering working on a particular functionality or aspect of the digital services platform, as described above. The deployment code portion may be determined by receiving or intercepting an indication that the deployment code portion is or has been pushed out to production environment 270. In some embodiments, the deployment code portion may be determined by receiving or intercepting a request to push the deployment code portion to production environment 270 or otherwise determining that a request to push the deployment code portion to the production environment 270 has been sent and/or received, as described above.

[0113] As depicted in Figure 4, the first code portion may be determined from the codebase 410 of the development environment 280, as described above in step 310. The set of second code portions may be determined from the codebase 410, as described in step 315. Determination of the first code portion and the set of second code portions is depicted in Figure 4 by arrow A. The canary system 210 may, upon receiving the first code portion and the set of second code portions, perform step 320 to determine the set of code portion combinations.

[0114] The deployment code portion or the request to push the deployment code portion may be determined by, or otherwise sent to, the code distribution server 250, as depicted by arrow B of Figure 4. In some embodiments, and as indicated by arrow C, the canary system 210, according to step 325, may determine that the deployment code portion is being pushed or has been pushed to the production environment 270 or a request to push the deployment code portion has been communicated to the code distribution server 250. Subsequent to determining that the deployment code is/has been

pushed or a request to push has been communicated, canary system 210 may, as depicted by arrow C, determine the deployment code portion. In some embodiments, canary code system 210, subsequent to determining that the deployment code is/has been pushed or a request to push has been communicated, may determine the deployment code portion from codebase 410, as indicated by arrow A.

[0115] Subsequent to determining the deployment code portion, the canary system 210 determines the candidate code portion, from the set of code portion combinations, according to step 325. As depicted by arrow C of Figure 4, the canary system 210 may communicate the candidate code portion to the code distribution server 250 for distribution to production environment 270. Code distribution server 250 distributes the deployment code portion to standard production servers 422, as depicted by arrow D. The code distribution server 250 distributes the candidate code portion to canary deployment server 425, as depicted by arrow E.

[0116] It will be appreciated by persons skilled in the art that numerous variations and/or modifications may be made to the above-described embodiments, without departing from the broad general scope of the present disclosure. The present embodiments are, therefore, to be considered in all respects as illustrative and not restrictive.

CLAIMS:

1. A computer implemented method comprising:
 - determining a first code portion;
 - determining one or more second code portions;
 - determining one or more code portion combinations by combining the first code portion with each of the second code portions;
 - responsive to determining that a deployment code portion is being pushed to a plurality of computing devices, determining the deployment code portion;
 - determining, based on the deployment code portion and the one or more code portion combinations, a candidate code portion that is indicative of the deployment code portion; and
 - causing the candidate code portion to be pushed to a sub-set of the plurality of computing devices.
2. The method of claim 1, comprising:
 - determining two or more second code portions; and
 - determining two or more code portion combinations.
3. The method of any one of the preceding claims, wherein determining the first code portion comprises receiving, from a code development computing device, code indicative of the first code portion.
4. The method of any one of the preceding claims, wherein determining the one or more second code portions comprises:
 - determining one or more pull requests; and
 - determining one or more code portions associated with each of the one or more pull requests.

5. The method of claim 3, wherein the pull request is indicative of a request to merge a branch of a code base associated with one of the second code portions with a main branch of the code base.
6. The method of claim 3 or claim 4, wherein the pull request comprises a canary identifier, and wherein the canary identifier indicates that the first code portion is permitted to be combined with the second code portion associated with the pull request.
7. The method of any one of claims 3 to 5, wherein determining one or more code pull requests comprises:
 - monitoring communications between one or more code development computing devices and one or more code distribution servers.
8. The method of any one of the preceding claims, wherein determining the one or more code change combinations further comprises:
 - prior to combining the first code portion with each of the second code portions, determining whether there is a conflict between the first code portion and one or more of the one or more second code portions.
9. The method of claim 8, wherein determining whether there is a conflict comprises determining whether the first code branch and one or more of the second code branches are associated with a same code section.
10. The method of claim 8 or claim 9, wherein if it is determined that there is a conflict, causing a warning to be sent.
11. The method of any one of the preceding claims, wherein determining that the deployment code portion is being pushed to a plurality of computing devices comprises:
 - receiving an indication from a code distribution server that the deployment code portion has been pushed or has been approved to be pushed.

12. The method of claim 11, wherein the indication is a web hook.
13. The method of claim 11 or claim 12, further comprising:
determining, from the indication, the second code portion or a universal resource indicator (URI) indicating the location of the second code portion.
14. The method of any one of claims 1 to 10, wherein determining that the deployment code portion is being pushed to a plurality of computing devices comprises determining that a request to push the deployment code portion to the plurality of computing devices has been sent.
15. The method of claim 14, wherein determining that a request to push a second code change to a plurality of computing devices has been sent comprises, monitoring traffic of a communications network to determine if the request to push a second code change has been communicated.
16. The method of claim 14 or claim 15, wherein determining that a request to push the deployment code portion to a plurality of computing devices has been sent further comprises:
subsequent to determining that the request to push the deployment code portion has been communicated, retrieving the deployment code portion from a computing device of a code development environment or a code distribution server.
17. The method of any one of the preceding claims, wherein determining the candidate code portion comprises:
comparing the deployment code portion or an indication of the deployment code portion to the one or more code portion combinations and determining which code portion combination is most similar to the deployment code portion.
18. The method of any one of the preceding claims, wherein determining one or more second code portions comprises:

receiving a notification that a second code portion has been deployed to a pre-production environment for preparation; and

determining, based on the notification, a copy of the second code portion associated with the notification.

19. The method of claim 18, wherein determining the one or more code portion combinations comprises:

subsequent to receiving the notification that that a second code portion has been deployed to a pre-production environment, and determining the copy of the second code portion associated with the notification;

combining a copy of the first code portion with the copy of the second code portion associated with the notification; and

submitting the combination of the first code portion with the copy of the second code portion to the pre-production environment.

20. The method of any one of the preceding claims, wherein the first code portion, each of the one or more second code portions, each of the one or more code portion combinations, the deployment code portion and the candidate code portion comprise one or more changes and/or additions to code of a codebase.

21. The method of any of the preceding claims, wherein the first code portion is indicative of code that is to be tested as part of a canary deployment.

22. A non-transitory machine-readable medium storing instructions which, when executed by one or more processors, cause a computing device to perform the method of any one of claims 1 to 21.

23. A system comprising:
one or more processors; and

memory comprising computer executable instructions, which when executed by the one or more processors, cause the system to perform the method of any one of claims 1 to 21.