**1) What is DevOps?**

In general, DevOps is the gray area between development (Dev) and operations (Ops) teams in a product development process. DevOps is a culture in which communication, integration, and collaboration in the product development cycle are emphasized. Thus, it eliminates the silos between software development and operations teams, allowing them to focus on rapid and continuous product deployment.

**1) What is continuous integration?**

Continuous integration (CI) is the process of automatically integrating code changes from multiple developers into a shared repository. Automated tests are utilized to verify and affirm the additional codes generate no conflict with the existing codebase. Ideally, code changes should be merged multiple times a day, at every commit, with the help of CI tools.

**3) What is continuous delivery?**

Continuous delivery (CD), together with CI makes a complete flow for deliverable code packages. In this phase, automated building tools are applied to compile artifacts (e.g., source code, test scripts, configuration files, and environments) and have them ready to be delivered to the end user. With that in mind, in a CD environment, new releases are just one click away from being published with fully functional features and minimal human intervention.

**4) What is continuous deployment?**

Continuous deployment takes CD to the next level by having new changes in code integrated and delivered automatically into the release branch. More specifically, as soon as the updates pass all stages of the production pipeline, they are deployed directly to the end-user without human intervention. Thus, to successfully utilize continuous deployment, the software artifacts must go through rigorously established automated tests and tools before being deployed into the production environment.

**5) What is continuous testing and its benefits?**

Continuous testing is the practice of applying automated tests early, gradually, and adequately in the software delivery pipeline. In a typical CI/CD workflow, builds are released in small batches. Therefore, it is impractical to manually perform test cases for each delivery. Automated continuous testing eliminates the manual steps and turns them into automated routines, which reduces human effort. That’s why automated continuous testing is essential for the DevOps culture.

Benefits of continuous testing:

* Ensures the quality and speed of builds
* Enables faster software delivery and a continuous feedback mechanism
* Detects errors as soon as they occur in the system
* Reduces business risks and evaluates potential problems before they become real problems

**6) What is version control and its uses?**

Version control (or source control) is a repository in which all changes in the source code are stored and managed at all times. Version control provides an operating history of code development that allows developers to rewind if they made any mistakes, compare versions, and finalize the best version of the code. All the code versions must be updated and checked into the version control, making it the single source of truth that is applicable for all environments.

**7) What is Git?**

Git is a distributed version control system that tracks changes in the code repository. Utilizing GitHub flow, Git revolves around a branch-based workflow that streamlines team collaboration as their projects progress over time.

**8) Why does DevOps matter? How does DevOps benefit teams in software delivery?**

In today’s digitized world, organizations must reinvent their product deployment systems to be more robust and flexible in order to keep up with the competition.

This is where the DevOps concept comes into the picture. DevOps plays a crucial part in generating mobility and agility to the entire software development pipeline, from idea to deployment and to the end-users. DevOps is the solution that drives a more streamlined and efficient process of continuously updating and improving products.

**9) Explain how DevOps is helpful for developers.**

In a world without DevOps, the workflow would start with developers building new codes, shipping, and integrating them, and then it’s the operation team’s duty to package and deploy the codes. After that, they have to wait for feedback. And if things go wrong, they have to go back and do it all over again to solve the bugs. In this process, there is exorbitant manual communication between different teams involved in the project.

Applying DevOps reduces developer tasks to building code only, as CI/CD practices consolidate and automate the remaining tasks. Combining the engineering and operation teams helps to create better communication and collaboration as processes become more transparent and accessible by all team members.

**10) Why has DevOps gained popularity in software delivery recently?**

DevOps has gained attention over the last few years primarily due to its ability to streamline the development, testing, and deployment processes for organizational operations and convert them into business value.

Technology is developing at a fast pace. Hence, organizations must adopt a new workflow – the DevOps and Agile approach – to streamline and stimulate their operations to stay ahead of the competition. DevOps’ features and benefits are reflected clearly through the successful application of the continuous deployment approach at both Facebook and Netflix, which boosted their growth without disrupting their ongoing operations.

**11) What are the benefits of CI/CD?**

The combination of CI and CD unifies all code changes into one single repository and runs them through automated tests, thus fully developing the product throughout all phases and making it ready for deployment at all times.

CI/CD enables organizations to roll out product updates as quickly, efficiently, and automatically as their customers expect them to be.

In short, a well-planned and well-executed CI/CD pipeline accelerates release rates and reliability and mitigates the code changes and defects of your product, resulting in much higher customer satisfaction.

**12) What are some benefits of continuous delivery?**

Manually releasing changes in code gives teams complete control of the product. In some cases, a new product release aligns with a bigger picture in the organization, for example, a promotion strategy with clear business purposes.

By automating repetitive and mundane tasks, IT professionals have more time to focus on improving the product without worrying about the integration progress.

**13) What are some benefits of continuous deployment?**

With continuous deployment, developers can focus solely on the product because their final task in the pipeline is to review pull requests and merge them to the master branch. This method enables frictionless deployment and shortens the deployment duration by releasing new features and fixes right after they have passed the automated tests.

Customers are the ones to evaluate the quality of each release. Bug fixes for new releases are easier to handle because each release is delivered in small batches.

**14) Define a typical DevOps workflow.**

A typical DevOps workflow can be organized into four stages:

* **Version control –** This is the stage in which the source code is stored and managed. The version control contains different versions of the code.
* **Continuous integration –** In this step, developers start to build components and have them compiled, validated, and tested with code review, unit testing, and integration testing.
* **Continuous delivery –** This is the next level of continuous integration, in which the release and testing processes are fully automated. CD ensures new releases are delivered quickly and sustainably to the end users.
* **Continuous deployment –** After the application has successfully passed all testing requirements, it is automatically deployed on the production server for release without any human intervention.

**15) What are the core operations of DevOps?**

The core operations of DevOps in terms of development and operation

**16) What are some precautions that teams need to consider before implementing DevOps?**

There are a few misconceptions about DevOps practices that could potentially lead to tragic failures when organizations try to apply this new approach:

* *DevOps is more than simply applying new tools and/or forming a new “department” and expecting it to work.* In fact, DevOps is considered a culture in which development and operations teams work side by side following a mutual framework.
* *Organizations do not need to define a clear vision for their DevOps practices.* Applying DevOps initiatives is a notable change to both development and operation teams. Therefore, having a clear roadmap, goals, and expectations for integrating DevOps in your organization will eliminate confusion and provide clear guidelines from the start.
* After the DevOps practices have been applied across the organization, the management team needs to establish and nurture a continuous learning and improvement culture. Failures and problems in the system should be treated as a valuable medium for teams to learn from mistakes and prevent those mistakes from recurring.

**17) What role does the SCM team play in DevOps?**

Software configuration management (SCM) is the practice of tracking and keeping records of the development environment, including all the changes and adjustments made in the operating system.

In DevOps, SCM is built as codes under the umbrella of *Infrastructure as Code* practice.

SCM simplifies tasks for developers as they no longer have to manually manage the configuration processes. This process is now constructed in a machine-readable form and is automatically duplicated and standardized.

**18) What role does the Quality Assurance (QA) team play in DevOps?**

As DevOps practices become a more desired culture in innovative organizations, QA teams’ responsibilities and relevance have shown signs of decline in today’s automated world.

However, this can be considered a myth. The rise of DevOps is not equal to the end of QA roles. It only means that their working environment and required expertise are changing. Therefore, their main focus should be to professionally evolve in order to keep up with this maturing trend.In DevOps, QA teams play a strategic role to ensure the stability of continuous delivery practices as well as perform the exploratory testing tasks that automated and repetitive tests cannot fulfill. Their insights in assessing the test and detecting the most valuable tests still play a crucial role in mitigating bugs in the last steps of a release.

**19) What are some tools used for DevOps? Describe your experience working with any of these tools.**

Throughout a typical DevOps lifecycle, there are various tools to support different phases of product development. The most common tools for DevOps can be divided into six key phases:

* **Continuous development:** Git, SVN, Mercurial, CVS, Jira
* **Continuous integration:** Jenkins, Bamboo, Hudson
* **Continuous delivery:** Nexus, Archiva, Tomcat
* **Continuous deployment:** Puppet, Chef, Docker
* **Continuous monitoring:** Splunk, ELK Stack, Nagios
* **Continuous testing:** Selenium, Katalon Studio

**19) How do you apply change management in DevOps practices?**

The typical change management approach needs to be appropriately integrated with DevOps’s modern practices. The first step is to centralize changes into a single platform to streamline the change, problem, and incident management processes.

Next, businesses should establish high transparency standards to make sure everyone is on the same page and assure accuracy over internal information and communication.

Stratifying the upcoming changes and establishing robust policies will help minimize risks and shorten the timeline for change. Lastly, organizations should apply automation into their process and integrate it with their DevOps software.

**21) What are some core components of CI/CD?**

A stable CI/CD pipeline requires a repository management tool that acts as a version control system so that developers can keep track of changes in the software build.

In the version control system, developers can also collaborate on a project, compare between versions, and undo any mistakes they make, thus mitigating disruption to all team members.

Continuous testing and automated testing are the two most pivotal keys to successfully establishing a seamless CI/CD pipeline. Automated testing must be integrated into all product development phases (including unit, integration, and system testing) to cover all functionalities such as performance, usability, performance, load, stress, and security.

**22) What are some common practices of CI/CD?**

Here are some best practices for establishing an efficient CI/CD pipeline:

* Develop a DevOps culture
* Implement and utilize continuous integration
* Deploy to every environment the same way
* Fail and restart your pipeline
* Apply version control
* Include the database in the pipeline
* Monitor your continuous delivery pipeline
* Get your CD pipeline flowing

**23) When is the best time to implement CI/CD?**

The transition to DevOps requires a complete reshaping of your software development culture, including the workflow, organizational structure as well as infrastructure. Therefore, organizations must prepare themselves for a major change when implementing DevOps.

**24) What are some common CI/CD servers?**

* Visual Studio

Visual Studio (VS) supports a fully developed DevOps system with agile planning, source code control, package management, testing and release automation, and continuous monitoring.

* TeamCity

TeamCity is an intelligent CI server that offers framework support and code coverage without any extra plugin install needed and no mods to build scripts.

* Jenkins

A self-contained CI server, it supports collaboration between development and operation teams with a shared pipeline and error tracking function. It can also be combined with hundreds of dashboard plugins.

* GitLab

GitLab’s users can customize the platform for efficient continuous integration and deployment. GitLab helps CI/CD teams to accelerate in-code shipping, errors identity, and program recovery rollout.

* Bamboo

Bamboo is a continuous integration server for product release management automation. Bamboo keeps track of all deployments on all tools and communicates errors in real-time.

**25) Describe an efficient workflow for continuous integration.**

A successful workflow when implementing continuous integration encompasses the following practices:

* Implement and maintain a repository for the project’s source code
* Automate the build and integration
* Make the build self-testing
* Commit changes daily to the baseline
* Build all commits added to the baseline
* Keep the builds fast
* Run tests in a clone of the production environment
* Make it easy to get the latest deliverables
* Make build results easy to be monitored by everyone
* Automate deployment

**26) What are some key differences between Agile and DevOps?**

Basically, DevOps and Agile are complementary to each other. Agile focuses more on the values and principles of developing new software and managing a complex process in a more effective way. Meanwhile, DevOps is all about enhancing the communication, integration, and collaboration between different teams, which comprises the developer and operation teams.

It takes both Agile and DevOps methods to form a product development life cycle that works seamlessly: Agile principles help shape and steer the development toward the right path and DevOps utilizes the tools to make sure the product is fully delivered to the customers.

**27) What are some differences between continuous integration, continuous delivery, and continuous deployment?**

CI is a practice in which code versions are continuously integrated into a shared repository. This practice ensures that new codes are automatedly tested and that errors are quickly detected and fixed.

Continuous delivery takes CI one step further by ensuring that after integration, the code base is ready to be released any time, within just a push of a button. Therefore, CI is considered a prerequisite for continuous delivery, the other essential part of the CI/CD pipeline.

As for continuous deployment, no manual step is required. Once the codes have passed the tests, they are automatically pushed to the production environment.

All three of these components (continuous integration, continuous delivery, and continuous deployment) are essential phases of implementing DevOps.

On one hand, continuous delivery is more suitable for applications that already exist with active users so that things can flow a bit slower and be more tuned. On the other hand, if you plan to release brand new software and want the whole process to be fully automated, continuous deployment is a more suitable choice for your product.

**28) What are some fundamental differences between continuous delivery and continuous deployment?**

Under continuous delivery, the codes in the main branch are always ready to be deployed manually. This practice allows the development team to decide when to release new changes or features to best benefit the organization.

Meanwhile, continuous deployment will have all updates and fixes in the code automatically deployed to the production environment immediately after the testing phase and without any human intervention.

**29) What are the differences between continuous integration and continuous delivery?**

Continuous integration helps ensure that software components work closely together. Integration should be done frequently; preferably on an hourly or daily basis. Continuous integration helps increase the frequency of code commits and reduces the complexity of connecting code from multiple developers. Eventually, this process reduces the chances for non-compatible code and redundant efforts.

Continuous delivery is the next step in the CI/CD pipeline. As the code is integrated into the shared repository continuously, continuously testing that code is made possible. There are no gaps while waiting for code to be completed before testing can occur. This ensures as many bugs are found as possible to then continuously deliver code to production.

**30) What are the differences between DevOps and continuous delivery?**

DevOps is more of an organizational and cultural approach that promotes collaboration and communication between both the engineering and operation teams.

Meanwhile, continuous delivery is an essential factor that contributes to the success of implementing DevOps into the product development workflow. Continuous delivery practices help to make new releases more reliable and establish a more seamless and shorter process.

The key purpose of DevOps is to effectively combine the Dev and Ops roles, remove all silos, and deliver business goals independent of continuous delivery practices.

Alternatively, continuous delivery works best when there is already a DevOps process in place. Thus, it amplifies collaboration and streamlines the unified product development cycle of the organization.

**31) What are the differences between Agile, Lean IT, and DevOps?**

Agile is a methodology that focuses solely on the development of software. Agile aims to iterate development, establish continuous delivery, shorten feedback loops, and improve team collaboration throughout the software development life cycle (SDLC)

Lean IT is a methodology that aims to streamline the value stream of the product development cycle. Lean focuses on eliminating unnecessary processes that do not add value and creating a flow to optimize the value streams

DevOps focuses on both the development and deployment — the Dev and Ops of the product development process. Its goal is to effectively unite the roles between automation tools and IT professionals for a more streamlined and automated process



3**2. Explain Continuous Integration, Continuous Delivery, and Continuous Deployment.**

* **Continuous Integration (CI):** Continuous integration (CI) is the practice of developers regularly integrating their code changes into a repository. Integration may take place several times a day and is verified by automated tests and a build process. As a result, integration challenges can be avoided, bugs can be found early in the development cycle, fixed, and tested iteratively. Every time new commits are integrated into the main branch, continuous integration emphasizes testing automation to make sure the application is not broken.
* **Continuous Delivery (CD):** In continuous delivery, all code changes are automatically deployed to the test and/or production environments after the build is complete. Feature additions, configuration changes, and error fixes are some examples of changes. By automating the delivery of new code to users, CD ensures a safe, quick, sustainable process. The CD process also involves some additional checks, like performance tests in production. With Continuous Delivery, deployments are predictable and scheduled regularly.
* **Continuous Deployment (CD):** The most critical stage of the pipeline is continuous deployment. By following this practice, you will be able to release all changes that have passed all stages of the production pipeline to your customers on time. Code changes can be made live much more quickly at this stage because there is little human interaction. In addition, continuous deployment allows you to accelerate your feedback loop with your customers and relieve pressure on your team since "release days" are no longer needed. Minutes after finishing their work, developers see their work go live.

**33. Explain the benefit of the CI/CD Pipeline.**

CI/CD brings certain benefits to your organization that will help you decide whether to go ahead with it. Below are some of these benefits:

* CI/CD makes it possible to quickly integrate small code changes and test the changes easily.
* With CI/CD setup, fault isolation is easier and faster. In fault isolation, the system is monitored, the fault occurrence time is determined, and the fault location is determined.
* With CI/CD, MTTR (mean time to resolution) is reduced since changes are smaller and easier to implement. CI/CD setup increases the reliability of the unit and integration testing.
* When CI/CD is set up, small code changes are merged very quickly and the code is ready for production very soon.
* CI improves transparency by detecting early-stage failures like build failures, merge issues, integration test failures, etc.
* CD facilitates the release of code to end-users.

**34. Does CI/CD require any programming knowledge?**

As far as CI/CD goes, it does not require any programming language or scripting language to be used. It is not necessary to use any programming or scripting language when you use a GUI-based tool like Azure DevOps (ADO). The use of ARM templates in Azure DevOps requires scripting knowledge. Therefore, it depends on the tools and different ways of setting up CI/CD.

**35. Can you explain the Git branch?**

The Git branch is essentially a separate line of development that can be used for working on a particular feature, usually during development. The use of branches allows developers to code without interfering with the work of other team members.

**36. Describe the build stage.**

The build stage is the first phase of the CI/CD pipeline, and it automates a lot of the steps that a typical developer goes through, such as installing tools, downloading dependencies, and compiling a project. Aside from building code, build automation involves the use of tools to verify that the code is safe and compliant with best practices. In this stage, the buildability and testability of the application are validated.

**37. What does containerization mean?**

As the term implies, containerization entails packaging together software code along with all the necessary components, such as frameworks, libraries, and other dependencies, in their own container. Among the advantages of containerization is that a container can be viewed as a fully packaged computing environment that can be transported in one piece.

**38. Explain trunk-based development.**

The trunk-based development approach ensures software remains up-to-date by integrating small, frequent updates into the main branch or a core "trunk". As a result of its ability to streamline merging and integration phases, it can be used to achieve CI/CD and to increase the speed and efficiency of the delivery of software and the efficiency of organizations. It is a branching model that consists of most of the work happening in a single trunk (also known as the trunk, master, or main). Each developer in the team merges their changes into the trunk on a daily basis. The reason why trunk-based development is popular is that it simplifies version control. This model minimizes merge conflicts due to the trunk's single source of truth.

**39. Can a branch live for a long time?**

Continuous integration follows trunk-based development practices, which means branches should be short-lived. Branch durations should be kept to a minimum of a few hours and a maximum of a day.

**40. What is the difference between a hosted and a cloud-based CI/CD platform?**

Hosted and Cloud-based CI/CD platforms differ in the following ways:

| **Hosted CI/CD platform** | **Cloud-based CI/CD platform** |
| --- | --- |
| It is necessary to manage a hosted CI server in the same way as any other server. Installation, configuration, and maintenance are all required before they can be used. | Cloud-based CI platforms, however, do not require maintenance. An organization can immediately start using CI/CD platforms in the cloud without installing or configuring anything. |
| Keeping the server secure requires updates and patches. | As the cloud is capable of providing all the machine power necessary, scalability is not an issue. |
| Additionally, failures in the CI server can halt development and deployment. | The cloud SLA (service-level agreement) guarantees the reliability of the platform. |

**41. In CI/CD, does security play an important role? How does it get secured?**

There are many factors that affect the security of CI/CD pipelines. These include:

* The importance of unit testing cannot be overstated when it comes to the testing of multiple unit-testable distributed components. It is therefore important to unit test your code properly.
* Static analysis security testing (SAST) scans your code for security vulnerabilities and the libraries you use. To ensure SAST scanning, all modern tools integrate well with the CD pipeline.
* DAST (dynamic analysis security testing) is a tool for securing your application by dynamically scanning for security vulnerabilities. It simulates the actions of an attacker by performing the tests outside the application.

**42. What are some of the deployment strategies?**

* **Regular release/deployment:** Through a single release, the software is made available to the public.
* **Canary releases:** These releases are intended to reduce the risk of failure by exposing a small fraction of the user base (around 1%) to the release. As part of a canary release, developers gradually transition users to the new release in a controlled manner.
* **Blue-green releases:** Essentially, it involves running two instances of an application simultaneously; one is the current stable version, and the other is the most recent version. An immediate switch is made from the old version to the new one. If there is a problem, users can immediately revert to the previous version, which is safer than regular or big-bang releases.

**43. What do you mean by Rolling Strategy?**

Rolling deployments update running instances of an application with new releases as they are released. The process involves replacing old versions of an application over time with new versions of the application by replacing the entire infrastructure on which the application is run.

**44. What is TDD?**

[Test-Driven Development](https://semaphoreci.com/blog/test-driven-development) (TDD) is a software design practice in which a developer writes tests before code. By inverting the usual order in which software is written, a developer can think of a problem in terms of inputs and outputs and write more testable (and thus more modular) code.

The TDD cycle consists of three steps:

1. **Red**: write a test that fails.
2. **Green**: write the minimal code that passes the test.
3. **Refactor**: improve the code, and make it more abstract, readable, and optimized.

TDD Cycle

**45. What is test coverage?**

Test coverage is a metric that measures how much of the codebase is covered by tests. A 100% coverage means that every line of the code is tested at least by one test case.

**46. Does test coverage need to be 100%?**

No. There’s a myth that 100% coverage means that the code is bug-free. This is false; no amount of testing can guarantee that. Attempting to reach full test coverage is considered bad practice because it leads to a false sense of security and extra work when code needs to be refactored.

**47. How can you optimize tests in CI?**

* Breaking large tests into smaller units.
* Removing obsolete tests.
* Refactoring tests to have fewer dependencies.
* [Parallelizing tests](https://semaphoreci.com/blog/revving-up-continuous-integration-with-parallel-testing).

**48. What’s the difference between end-to-end testing and acceptance testing?**

End-to-end usually involves testing the application by using the UI to simulate user interaction. Since this requires the application to run in a complete production-like environment, end-to-end testing provides the most confidence to developers that the system is working correctly.

[Acceptance testing](https://semaphoreci.com/blog/the-benefits-of-acceptance-testing) is the practice of verifying acceptance criteria. Acceptance criteria is a document with the rules and behaviors that the application must follow to fulfill the users’ needs. An application that fulfills all acceptance criteria meets the users’ business needs by definition.

The confusion stems from the fact that acceptance testing implements the acceptance criteria verification with end-to-end testing. That is, an acceptance test consists of a series of end-to-end testing scenarios that replicate the conditions and behaviors expressed in the acceptance criteria.

#### **48. Could you explain the process of deployment?**

The deployment process flow consists of 5 steps:

* Planning
* Development
* Testing
* Deploying
* Monitoring

#### **49 What is the difference between build and deploy?**

Build and deploy are testing terms with important meanings in the IT field. “Build” means consolidating and combining a set of executable code for testing. “Deploy” means injecting that set of executable code into a particular software environment to test it.

#### **50. Can you elaborate on the release flow?**

Flow is a trunk-based development approach in which the master branch is deployed and developed. Branches are created off the master branch when there are special needs, but the branches do not serve as deployment targets. Here a pull request merges a branch back into the master branch.

#### **51. What is the most important deployment step?**

Systematic communication is an important element of deployment management. Without communication, many problems can arise, including scheduling conflicts and misunderstandings about the scope of a deployment. Post-deployment check-ins with decision-makers can help your team arrive at timely and accurate decisions, allowing you to avoid many common deployment issues.

52) What deployment strategy you used?

Recreate, Green-blue and canary.