04/16/2023 **Devops questions and answers**

**1. What is meant by DevOps, and what is its purpose?**

DevOps is a set of practices, tools, and philosophies used to improve communication and collaboration across different teams when delivering new software. In short, it bridges the gap between developers and IT staff.

The goal of DevOps is to integrate automation and monitoring at all stages of the software development lifecycle. This improves the speed at which new systems and applications can be delivered.

**2. What are the main types of DevOps tools?**

Organizations use several different types of DevOps tools, including:

* Version control tools, such as Git
* Continuous integration tools, such as Jenkins
* Configuration management tools, such as Puppet
* Containerization tools, such as Docker
* Cloud tools, such as AWS
* Monitoring tools, such as Nagio2s.

**3. What are some of the key skills of a DevOps engineer?**

DevOps encompasses several different processes within the software development lifecycle. As such, DevOps engineers must have a wide range of soft and hard skills, including:

* A solid understanding of DevOps
* [Communication](https://www.testgorilla.com/test-library/situational-judgment-tests/communication-test/) and teamwork skills
* [Programming skills](https://www.testgorilla.com/test-library/test-type/programming-skills-tests)
* Software security skills
* Cloud technology skills

**4. What are the differences between DevOps and Agile?**

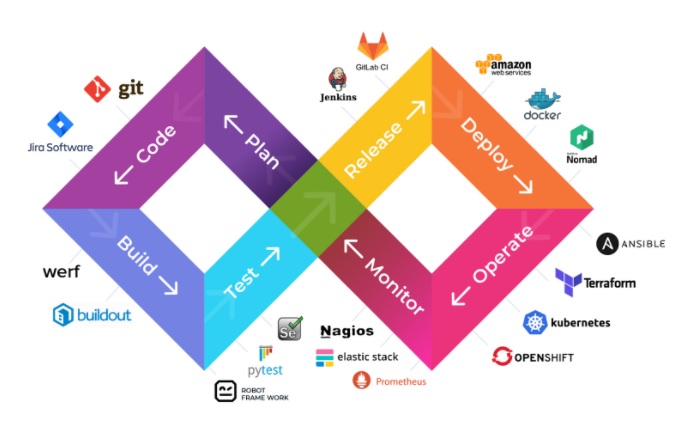
DevOps arose largely as an alternative to Agile workflows like Scrum. The key differences include:

* DevOps brings together all teams involved in software development and maintenance, whereas Agile typically siloes specific tasks to different teams
* DevOps emphasizes consistency, stability, and automation, whereas Agile focuses on adaptability and iterative development
* DevOps encompasses the full flow of software from ideation to delivery and maintenance, whereas Agile generally ends with the completion of code.

**5. Can one consider DevOps as an Agile methodology?**

DevOps is not considered an Agile methodology, but it can be used in conjunction with Agile practices to improve the software development process. Agile methodology focuses on delivering small increments of working software frequently, while DevOps focuses on improving collaboration and communication between development and operations teams to increase efficiency and reduce the time it takes to release software to production. DevOps can be seen as a complementary approach to Agile, as it can help Agile teams to better achieve their goals of delivering software quickly and reliably.

**6. Explain the different phases in DevOps methodology?**



The various phases of the DevOps lifecycle are as follows:

* **Plan** – In this stage, all the requirements of the project and everything regarding the project like time for each stage, cost, etc. are discussed. This will help everyone in the team to get a brief idea about the project.
* **Code** – The code is written over here according to the client’s requirements. Here codes are written in the form of small codes called units.
* **Build** – Building of the units is done in this step.
* **Test** – Testing is done in this stage and if there are mistakes found it is returned for re-build.
* **Integrate** – All the units of the codes are integrated into this step.
* **Deploy** – code DevOps is deployed in this step on the client’s environment.
* **Operate** – Operations are performed on the code if required.
* **Monitor** – Monitoring of the application is done over here in the client’s environment.

**7. Which are the top DevOps tools? Which tools have you worked on? Same question as question2**

The most popular DevOps tools are mentioned below:

* Git: Version Control System tool
* Jenkins: Continuous Integration tool
* Selenium: Continuous Testing tool
* Puppet, Chef, Ansible: Configuration Management and Deployment tools
* Nagios: Continuous Monitoring tool
* Docker: Containerization tool

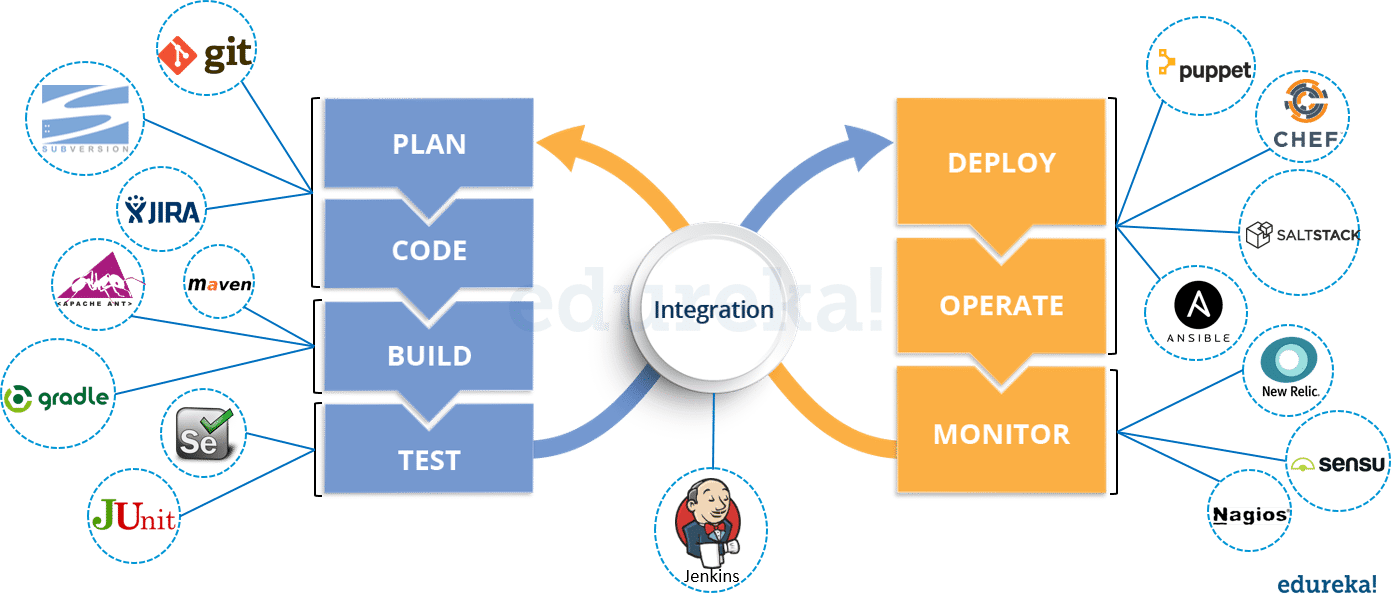
You can also mention any other tool if you want, but make sure you include the above tools in your answer. The second part of the answer has two possibilities:

1. If you have experience with all the above tools then you can say that I have worked on all these tools for developing good quality software and deploying those softwares easily, frequently, and reliably.
2. If you have experience only with some of the above tools then mention those tools and say that I have specialization in these tools and have an overview about the rest of the tools.

**8. How do all these tools work together?**

Given below is a generic logical flow where everything gets automated for seamless delivery. However, this flow may vary from organization to organization as per the requirement.

* Developers develop the code and this source code is managed by Version Control System tools like Git etc.
* Developers send this code to the Git repository and any changes made in the code is committed to this Repository.
* Jenkins pulls this code from the repository using the Git plugin and build it using tools like Ant or Maven.
* Configuration management tools like puppet deploys & provisions testing environment and then Jenkins releases this code on the test environment on which testing is done using tools like selenium.
* Once the code is tested, Jenkins send it for deployment on the production server (even production server is provisioned & maintained by tools like puppet).
* After deployment It is continuously monitored by tools like Nagios.
* Docker containers provides testing environment to test the build features.



**9. What is the role of configuration management?**

Configuration management (CM) is the process in which software systems are automated, updated, monitored, and managed. It helps reduce the risk of unexpected system failures and offers greater agility for personnel working across the DevOps strategy.

**10. What is the role of continuous integration?**

Continuous integration (CI) is the process in which the integration of code changes into a software project is automated. It allows developers to merge code changes into a single repository whenever a new change is made, thus improving collaboration and transparency.

**11. What is the role of continuous testing?**

Continuous testing involves regular testing of software at every stage of the development lifecycle to ensure any bugs are fixed rapidly. It provides continuous feedback so that developers can evaluate software quality throughout the delivery process.

**12. What is the role of continuous monitoring?**

Continuous monitoring refers to the regular monitoring of software at every stage of the development lifecycle to ensure the performance, reliability, and compliance of the application and infrastructure. This provides transparency and facilitates the early detection of any issues.

**13.  What are the advantages of DevOps?**

For this answer, you can use your past experience and explain how DevOps helped you in your previous job. If you don’t have any such experience, then you can mention the below advantages.

Technical benefits:

* Continuous software delivery
* Less complex problems to fix
* Faster resolution of problems

Business benefits:

* Faster delivery of features
* More stable operating environments
* More time available to add value (rather than fix/maintain)

**14. Mention some of the core benefits of DevOps?**

* Faster development of software and quick deliveries.
* DevOps methodology is flexible and adaptable to changes easily.
* Compared to the previous software development models, confusion about the project is decreased due to increased product quality.
* The gap between the development team and operation team is bridged. i.e, the communication between the teams has been increased.
* Efficiency is increased by the addition of automation of continuous integration and continuous deployment.
* Customer satisfaction is enhanced.

**15. What do you know about Jenkins?**

Jenkins is the most popular continuous integration and continuous delivery (CI/CD) tool on the market today. Pretty much all DevOps teams rely on Jenkins to automate parts of software development related to the build, test, and deployment of applications.

**16. How are application development and infrastructure different? Application development consists of the following core operations:**

* Coding
* Unit testing
* Packaging
* Deployment

Infrastructure consists of:

* Provisioning
* Configuration
* Monitoring
* Deployment

**17. Explain your understanding and expertise on both the software development side and the technical operations side of an organization you have worked with in the past.**

For this answer, share your past experience and try to explain how flexible you were in your previous job. You can refer the below example:  
DevOps engineers almost always work in a 24/7 business-critical online environment. I was adaptable to on-call duties and was available to take up real-time, live-system responsibility. I successfully automated processes to support continuous software deployments. I have experience with public/private clouds, tools like Chef or Puppet, scripting and automation with tools like Python and PHP, and a background in Agile.

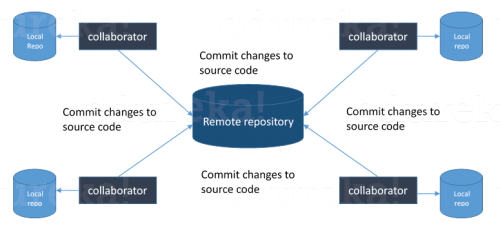
### 18.  What do you know about Git?

Git is a popular open-source version control system used to track changes in the source code, create multiple scripts, and, ultimately, improve collaboration between developers. It’s a critical tool in the coding stage of the software development lifecycle.

**19.  What is Git?**

I will suggest that you attempt this question by first explaining about the architecture of git as shown in the below diagram. You can refer to the explanation given below:

* Git is a Distributed Version Control system (DVCS). It can track changes to a file and allows you to revert back to any particular change.
* Its distributed architecture provides many advantages over other Version Control Systems (VCS) like SVN one major advantage is that it does not rely on a central server to store all the versions of a project’s files. Instead, every developer “clone” a copy of a repository I have shown in the diagram below with “Local repository” and has the full history of the project on his hard drive so that when there is a server outage, all you need for recovery is one of your teammate’s local Git repository.
* There is a central cloud repository as well where developers can commit changes and share it with other teammates as you can see in the diagram where all collaborators are committing changes “Remote repository”.



**20. What are the key differences between git fetch and git pull?**

* Git fetch and git pull are two important commands in GitHub that are often confused with each other.
* Git fetch retrieves the latest data from the remote repository, but without integrating this data into the working branches. Git pull, on the other hand, retrieves *and*updates local branches with the new information from their corresponding remote branches.

**21. What are the key differences between git merge and git rebase?**

* Git merge and git rebase are two commands in GitHub used to integrate changes from one branch into another. However, they integrate these changes in different ways.
* A git merge creates a new commit from the head branch, whereas a git rebase rewrites the changes of one branch onto another without creating a new commit.

**22. What is Version control?**

This is probably the easiest question you will face in the interview. My suggestion is to first give a definition of Version control. It is a system that records changes to a file or set of files over time so that you can recall specific versions later. Version control systems consist of a central shared repository where teammates can commit changes to a file or set of file. Then you can mention the uses of version control.

Version control allows you to:

* Revert files back to a previous state.
* Revert the entire project back to a previous state.
* Compare changes over time.
* See who last modified something that might be causing a problem.
* Who introduced an issue and ` when?

**23. What are the benefits of using version control? ANSWER1**

I will suggest you to include the following advantages of version control:

* With Version Control System (VCS), all the team members are allowed to work freely on any file at any time. VCS will later allow you to merge all the changes into a common version.
* All the past versions and variants are neatly packed up inside the VCS. When you need it, you can request any version at any time and you’ll have a snapshot of the complete project right at hand.
* Every time you save a new version of your project, your VCS requires you to provide a short description of what was changed. Additionally, you can see what exactly was changed in the file’s content. This allows you to know who has made what change in the project.
* A distributed VCS like Git allows all the team members to have complete history of the project so if there is a breakdown in the central server you can use any of your teammate’s local Git repository.

### 24. What are the benefits of version control? ANSWER2

Version control systems improve the efficiency of coding. This is achieved through:

* **Traceability**: Version control tracks changes to code from all developers, providing a clear history that helps improve the functionality of the software
* **Branching**: Version control allows developers to work on code independently without impacting contributions from other collaborators
* **Error reduction**: Version control helps detect the root cause of software bugs as well as any duplications for easy removal.

### ****25. Which VCS tool you are comfortable with?****

You can just mention the VCS tool that you have worked on like this: “I have worked on Git and one major advantage it has over other VCS tools like SVN is that it is a distributed version control system.”  
Distributed VCS tools do not necessarily rely on a central server to store all the versions of a project’s files. Instead, every developer “clone” a copy of a repository and has the full history of the project on their own hard drive.

**26.  What is meant by branching?**

Branching is a technique used by developers within version control systems, whereby the source code is copied to make two versions that are developed separately. These branches can then be developed independently without affecting the code base, thus promoting collaboration.

**27. What are the two main types of branching strategies? There two main branching strategies available to developers are:**

* Release branching, which creates a branch for a potential new release
* Feature branching, which creates a branch for specific features or tasks. We also have Task Branching (each task is implemented on its own branch with the task key included in the branch name.)

**28. What are the main phases in the DevOps lifecycle?**

The software development lifecycle consists of planning, coding, build, testing, release, deployment, and monitoring. When applied to DevOps, there are four key phases:

* Continuous Integration, which includes the coding and build stages
* Continuous Delivery, consisting of the testing and release stages
* Continuous Deployment, which includes the release and deployment stages
* Continuous Feedback, consisting of the monitoring stage.

**29. What do you know about Puppet?**

Puppet is a widely used open-source software configuration management tool that supports automated testing, continuous integration, and continuous delivery. It’s specifically designed for Windows and Linux systems.

Puppet’s infrastructure consists of the main server environment, which stores all codes, and the client.

30. **What are the key differences between continuous delivery and continuous deployment?** Continuous delivery and continuous deployment are two parts of continuous integration:

* Continuous delivery deploys all code changes to a testing or production environment
* Continuous deployment automatically releases new changes to customers.

**31. What are the key differences between continuous testing and automation testing?**

* Automated testing is the process of automating a set of tasks to improve speed and reduce error.
* Continuous testing, on the other hand, encompasses a wider scope of applications. It focuses on business risks and sets out to achieve continuous improvements that mitigate those risks.

**32. What do you know about Docker?**

Docker is a containerization tool used by DevOps teams during the continuous deployment stage. It packages applications and all of their constituent parts inside software containers, which are then ready for deployment across different operating systems.

**33. What is meant by CAMS?**

CAMS is an acronym used to describe four of the key DevOps principles. These include:

* **Culture**: DevOps is upheld by a culture of collaboration and transparency
* **Automation**: DevOps establishes repeatable (or automated) systems to reduce errors and save time
* **Measurement**: DevOps relies on continuous performance tracking and feedback to improve efficiency
* **Sharing**: DevOps teams share all information and feedback, including problems.

**34. What are some of the best KPIs for evaluating DevOps performance?**

There’s a wide range of KPIs in the DevOps field. Some of the most effective ones include:

* Deployment frequency: Measures how often new features are launched
* Change volume: Measures the extent to which the code is changed in new deployments
* Deployment failure rate: Measures how often new deployments lead to outages or other issues
* Deployment time: Measures the time taken to roll out new deployments
* Mean time to recovery: Measures the average time taken to recover from a system failure.

### 35. Name three important DevOps KPIs

* **Lead time for changes:** It measures the time taken from committing a change to code repository to the time it becomes available in production.
* **Deployment frequency:** It measures the number of times changes are deployed to production in a given period of time.
* **Mean time to recover (MTTR):** It measures the average time taken to recover from a service disruption or failure.

### ****36. How does AWS contribute to DevOps?****

AWS [Amazon Web Services] is one of the famous cloud providers. In AWS DevOps is provided with some benefits:

* **Flexible Resources:** AWS provides all the DevOps resources which are flexible to use.
* **Scaling:** we can create several instances on AWS with a lot of storage and computation power.
* **Automation:** Automation is provided by AWS like CI/CD
* **Security:** AWS provides security when we create an instance like IAM

### 37. What do you know about Nagios?

Nagios is a widely used open-source monitoring system that runs periodic checks on the critical parameters of servers, networks, and applications. It keeps DevOps teams updated on things like memory, disk usage, log files, and microprocessor load. Supporting continuous monitoring, Nagios alerts technical staff of issues before they materialize and impact the end user.

### 38. What are the key differences between asset management and configuration management?

* Configuration management refers to the management of assets as entities within the software development pipeline. This involves reviewing the accuracy and reliability of different configuration items, as well as the relationships they have with each other.
* Asset management, on the other hand, reviews these assets from a financial perspective. It tracks items from the point of acquisition to disposal, to determine whether they offer sufficient economic value to the organization.

**39. What are anti-patterns? Can you name a couple?**

Anti-patterns are ideas that are counter-productive to the DevOps philosophy. They’ll often fix a short-term problem at the expense of a long-term goal. Some examples include:

* DevOps is a process rather than a culture
* DevOps is driven either by development or IT operations, rather than a combination of both
* Agile equals DevOps?
* We need a separate DevOps group
* Devops will solve all our problems
* DevOps means Developers Managing Production
* DevOps is Development-driven release management
* DevOps is not development driven.
* DevOps is not IT Operations driven.
* We can’t do DevOps – We’re Unique
* We can’t do DevOps – We’ve got the wrong people.

**40. What is the role of cloud computing in DevOps?**

Cloud computing provides a centralized, scalable communication platform for DevOps teams at each stage of the software development lifecycle. It allows team members to collaborate more quickly and closely.

**41. Why has DevOps become famous?** As we know before DevOps there are two other software development models:

* Waterfall model
* Agile model

### 42. What do you know about Ansible?

Ansible is a popular DevOps automation tool used during the build, configuration, and management phases. By automating processes like testing and deployment, Ansible helps DevOps teams save time, reduce errors, and scale in pace with growing demand.

### 43. What is pair programming?

Pair programming is a popular programming technique whereby two developers work together on the same task, sharing a single computer. Typically, one developer will write the code, while the other will review each line of code as it is typed in.

### 44. What do you know about Chef?

Chef is a popular configuration management tool that supports continuous delivery by automating processes across several DevOps stages. Using the Ruby programming language, Chef translates system tasks into repeatable actions, known as recipes and cookbooks.

### 45. When are post-mortem meetings used?

Post-mortem meetings are used by DevOps teams typically between the release of a new iteration and the planning stage of the next one. Team members discuss the successes and failures of the previous project, reflecting on what can be improved next time.

### 46.What do you know about Selenium?

Selenium is a popular open-source framework for testing web applications. It’s widely used by DevOps teams, allowing them to implement automated testing without needing to use a formal test scripting language.

### 47. What are the benefits of cloud tools in DevOps?

Cloud computing tools like AWS and Azure support the CI/CD (continuous integration and continuous development) phases of the DevOps lifecycle. They allow DevOps teams to share code, track work, and deploy software remotely, across any platform.

**48. What are the best strategies for improving DevOps performance?**

The performance of a DevOps initiative can be improved in the following ways:

* Design a clear roadmap during the planning stage
* Use efficiency-based KPIs
* Use centralized storage
* Perform regression testing
* Focus on culture rather than processes
* Upgrade the DevOps infrastructure

**49. How can DevOps be implemented securely?**

Security practices should be incorporated into every step of the DevOps lifecycle. To implement them, teams can:

* Formalize cybersecurity policies within the team
* Use privileged access management
* Use secrets management
* Segment network access
* Automate security processes

**50. What is the “shift left to reduce failure” concept?**

The term “shift left to reduce failure”, or simply “shift left”, refers to efforts within DevOps teams to identify defects as early as possible in the software development lifecycle. Essentially, it involves regular testing at the start of the development pipeline rather than at the end.

**51. How does the blue-green deployment pattern work?**

The blue-green deployment pattern is a deployment strategy that aims to reduce new release risks. It involves running two separate production environments (blue and green), one for the current production system and the other for staging the new release.

Traffic is gradually shifted from the old production environment to the new release once testing is complete and software is deployed. The new release becomes live and, once the traffic has been completely transferred, the old production environment becomes idle.

**52. How does the canary deployment pattern work?**

The canary deployment pattern is a deployment strategy that aims to minimize the impact of potential defects in a new software release. It involves rolling out updates to a small subset of users before making them universally available.

The development team uses a router or load balancer to target individual routes with the new release. After launch, metrics are collected to evaluate the performance of the update, and a decision is made on whether the release is ready to be rolled out on a larger scale.

**53. What is infrastructure as code, and how is it implemented?**

* Infrastructure as Code (IaC) is the management of an application’s infrastructure through software (namely code) rather than manual processes.
* The infrastructure as code concept can be implemented at various stages of the DevOps lifecycle—version control, continuous integration, and automated testing—by writing code. This process facilitates more efficient, reliable, and secure changes to applications.

**54. What are the key differences between containerization and virtualization?**

* Containerization and virtualization are the two most popular methods for hosting applications in a computer system.
* Virtualization allows developers to run multiple operating systems on the hardware of a single physical server. Containerization enables developers to deploy multiple applications under the same operating system on a single virtual machine or server.

**55. How is regression testing implemented?**

* Regression testing assesses how an application behaves after a new change has been implemented. It should be performed between integration testing and user testing.
* Regression testing can be implemented by checking the original code after new changes have been made or by evaluating how updates affect performance. It can also be automated.

**56. What are the key differences between cloud and on-premises services in DevOps?**

Cloud and on-premises services are the two main data hosting pathways available to DevOps teams. With on-premises services, data is stored on in-house servers. With cloud services, data is hosted remotely by a third-party provider.

There are strengths and weaknesses to both strategies, which can be summarized as:

* Cloud services offer less security control over data and infrastructure, but they scale better, offer extra services, and typically incur lower costs.
* On-premises services come with large maintenance costs and unique security threats, but they provide greater control and customization scope.

**57. What is load balancing?**

Load balancing is the process of evenly distributing incoming network traffic across a group of backend servers. It occurs during the deployment phase of the DevOps lifecycle and is usually automated by developers.

**58. How can the DevOps troubleshooting process be optimized?**

Every DevOps team should have an established troubleshooting framework. Some of the best practices for troubleshooting are to:

* Set troubleshooting responsibilities within the team
* Gather relevant information including metrics, logs, and errors
* Hypothesize potential causes and tick them off one by one
* Closely monitor behavior after the issue has been fixed
* Report on issue resolution

**59. What are the three main types of cloud computing services?**

The three main types of cloud services are:

* Infrastructure-as-a-Service (IaaS), which provides internet-based access to storage and computing power, while the developer is responsible for managing everything else
* Platforms-as-a-Service (PaaS), which helps developers build web applications without needing to manage the underlying infrastructure, such as networks, servers, and storage
* Software-as-a-Service (SaaS), which hosts all components of the software application, including servers, storage, data, and the application itself

**60. What is the difference between monitoring and observability?**

* Monitoring is a process of tracking and data collection that helps DevOps teams better understand the current state of their systems.
* Observability, on the other hand, uses these findings to diagnose problems and actively resolve them through debugging.

**61. What are the key differences between centralized version control and distributed version control?**

With centralized version control, the server’s remote repository provides the latest code to client machines. With decentralized, or distributed, version control, the complete codebase is made available on each computer’s own local repository.