|  |  |
| --- | --- |
|  | **Sri Sivasubramaniya Nadar College of Engineering, Kalavakkam – 603 110**  **(An Autonomous Institution, Affiliated to Anna University, Chennai)**  Department of Information Technology  **UG**  **VII SEM**  **Continuous Assessment Test – I**  **Regulations – R2021** |

|  |  |  |  |
| --- | --- | --- | --- |
| **Degree**  B.E. / B. Tech. | B. Tech. | **Branch** | IT |
| **Semester** | VII | **Date of CAT** | 28.10.2024 |
| **Subject Code & Name** | **UIT2739 FULL STACK DEVELOPMENT** | | |
| **Time: 90 Minutes** | **Answer All Questions** | | **Maximum: 50 Marks** |

(K1: Remembering, K2: Understanding, K3: Applying, K4: Analyzing, K5: Evaluating)

|  |  |
| --- | --- |
| CO1 | Getting comfortable with Bash commands and VIM editor |
| CO2 | Use Git for SVC |
| CO3 | Use cloud, containers, and orchestration for [server setup](https://lms.ssn.edu.in/mod/resource/view.php?id=79232), security configurations, load balancing, and deployment |
| CO4 | Use sql/no-sql data and web services/API for data integration and interoperability |

**Part – A (4 × 2 = 8 Marks)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | KL | CO | PI |
| 1. | Write the grep command to display specific types of files. | K2 | CO1 | 1.1.2 |
| 2. | What is the difference between centralized and distributed version control? In what ways can version control support documentation and project management? | K1 | CO1 | 1.4.2 |
| 3. | What are the main differences between public, private, and hybrid clouds, and how do they impact cost, security, and scalability? | K2 | CO2 | 13.1.1 |
| 4. | How does HTTPS secure data? What does HTTPS stand for? | K2 | CO4 | 1.4.2  2.2.1  13.2.1 |

**Part – B (3×6 = 18 Marks)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  | KL | CO | PI |
| 5 | Explain the purpose of standard streams in Bash and how redirection is used in shell scripting. | | K1 | CO3 | 6.2.1  13.1.2 |
| 6 | Describe the function of SSH in server management. | | K3 | CO2 | 13.2.1  2.2.4 |
| 7 | Compare HTTP versions with a focus on performance, architecture, security, and practical applications | | K3 | CO1 | 1.4.1 |

**Part – C (2 × 12 = 24 Marks)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  | KL | CO | PI |
| 8. | A development team of five members is working on a large software project using Git. They are facing issues with merge conflicts and inconsistent code versions. What strategies can the team implement to minimize merge conflicts?How can the team effectively utilize Git branching to manage features and releases? How can version control systems be integrated into a continuous integration/continuous deployment (CI/CD) pipeline? | | K3 | CO2 | 1.4.2  14.2.2 |
| (Or) | | | | | |
| 9. | Write a shell script that accepts a list of filenames and counts the lines, words, and characters in each file. | | K3 | CO2 | 1.4.2  14.2.2 |
|  | | | | | |
| 10. | A former employee with excessive permissions was found to have accessed and leaked sensitive company data after they left the organization. How should the organization structure a permissions review to prevent such incidents? What steps should they take to implement the principle of least privilege across departments? What logs could provide evidence of unauthorized access, and what could a proactive monitoring system do to catch suspicious behavior in real time? How can the company use firewalls to restrict access to sensitive data on internal servers, and what role would port management play in limiting access? | | K3 | CO5 | 3.2.1  3.2.2  13.1.1 |
| (Or) | | | | | |
| 11. | A development team plans to use Docker as their container runtime environment to streamline application development and deployment. What is Docker’s role in the containerization ecosystem, and what core features make it useful for developers and DevOps teams? How should the team structure their workflow around Docker to ensure efficient image building, management, and deployment? | | K3 | CO5 | 3.2.1  3.2.2  13.1.1 |

|  |  |  |  |
| --- | --- | --- | --- |
| Prepared By  Course Instructor | Reviewed By  Panel member | Reviewed By  Course Coordinator | Approved By  HoD |
| Dr. N. Radha  Dr. K. R. Uthayan | Dr. S. Karthika  Dr. D. Preetha Evangeline | Dr. R. Srinivasan | Dr. A. Shahina |

|  |
| --- |
| **Review Comments by Panel Member 1** |
|  |
| **Review Comments by Panel Member 2** |
|  |
| **Review by Course Coordinator** |
|  |

1. find . -type f -name "\*.txt"
2. **Centralized Version Control (CVC)**: In CVC (e.g., SVN), there is one central repository, and all users work by accessing this central location. Users must be connected to this server to commit changes, which can create a bottleneck if the server goes down.

**Distributed Version Control (DVC)**: In DVC (e.g., Git), each user has a complete local copy of the repository. Users can work offline, commit changes locally, and push changes to a remote repository later. This structure avoids a single point of failure and improves fault tolerance.

1. Public Cloud: Managed by third-party providers and shared by multiple clients, public clouds are highly scalable and usually more cost-effective due to shared resources. Security is managed by the provider, but control is more limited.

Private Cloud: Dedicated to a single organization, often hosted on-premises or at a dedicated data center. Private clouds offer greater control and security but generally have higher costs and are limited in scalability to the organization’s available resources.

Hybrid Cloud: Combines public and private cloud environments, allowing organizations to keep sensitive data in a private setup while leveraging the public cloud for other resources. This approach balances cost, security, and scalability.

1. HTTPS stands for Hypertext Transfer Protocol Secure. It secures data through encryption using TLS (Transport Layer Security) or SSL (Secure Sockets Layer), ensuring that data transferred between the user’s browser and the server is protected from eavesdropping and interception. HTTPS also authenticates the server to confirm its identity, which helps prevent attacks like man-in-the-middle attacks.
2. Standard Input (stdin): Receives input data (e.g., keyboard input) and is represented by file descriptor 0. By default, stdin is connected to the terminal, allowing input from users.

Standard Output (stdout): Outputs data (e.g., command results) and is represented by file descriptor 1. By default, stdout is sent to the terminal screen.

command > output.txt # Overwrites the file

command >> output.txt # Appends to the file

Standard Error (stderr): Outputs error messages and is represented by file descriptor 2. Like stdout, stderr also defaults to the terminal but is used specifically for errors.

1. SSH (Secure Shell) is a protocol used for securely connecting to and managing remote servers over a network. It provides encrypted communication, ensuring that data transferred between the client and server remains confidential and secure. SSH enables users to:

Execute commands remotely: Admins can manage servers by running shell commands as if they were on the local machine.

Transfer files securely: Using tools like scp (secure copy) and sftp (SSH File Transfer Protocol), users can securely upload and download files.

Manage server configurations: Admins can edit configuration files, start and stop services, and monitor system status.

SSH’s encryption, strong authentication (such as using key pairs), and secure tunneling make it a fundamental tool for secure, remote server management.

1. HTTP/1.1

Performance: Introduced persistent connections, allowing multiple requests on the same connection. However, it’s limited by sequential request-response cycles, often requiring multiple connections.

Architecture: Single request-response per connection, leading to latency due to connection re-establishment.

Security: Can use HTTPS for encryption, but lacks built-in security measures.

Practical Applications: Still widely used in traditional web services, though its limitations in handling modern web traffic have led to slower page loads on complex sites.

HTTP/2

Performance: Introduced multiplexing, which allows multiple requests and responses over a single connection. Also uses header compression and server push, which reduces latency.

Architecture: Based on a binary protocol, which is faster and more efficient than HTTP/1.1's text-based protocol.

Security: Typically requires HTTPS, promoting more secure data transfer.

Practical Applications: Widely adopted by modern websites and applications for faster page loads, better performance, and reduced latency in data-heavy applications.

HTTP/3

Performance: Uses QUIC (Quick UDP Internet Connections) protocol, providing faster connection setup and data transfer, even in high-latency networks.

Architecture: Connectionless (uses UDP), which reduces handshakes and improves connection resilience.

Security: Has built-in encryption, providing end-to-end data security by default.

Practical Applications: Useful for mobile applications, streaming, and any application needing low-latency performance, especially where network conditions may be unstable.

1. Frequent Commits and Pulls: Team members should commit changes frequently and pull the latest code from the main branch before starting new work. Regularly syncing with the latest code minimizes conflicts when merging.

Feature Branches: Each team member should work on separate feature branches instead of directly on the main branch. This isolates changes, reducing the likelihood of conflicts with other developers’ code.

Smaller, Incremental Changes: Breaking down work into smaller, manageable tasks can help prevent large, conflicting changes. Each feature or bug fix should be a separate commit or small set of commits.

Clear Code Ownership: Assign specific files or areas of the codebase to certain team members when possible, reducing overlap and the chance of conflicting changes.

Code Review: Regular code reviews catch potential conflicts and encourage team members to discuss and resolve issues proactively.

Rebasing Instead of Merging: Team members can rebase their branches against the main branch before merging to avoid complex conflict resolutions. However, rebasing should be used cautiously, as it rewrites commit history.

1. #!/bin/bash

# Check if at least one filename is provided

if [ "$#" -eq 0 ]; then

echo "Usage: $0 filename1 [filename2 ... filenameN]"

exit 1

fi

# Loop through each filename passed as an argument

for file in "$@"; do

# Check if the file exists and is readable

if [ -r "$file" ]; then

# Use wc to count lines, words, and characters

echo "File: $file"

wc "$file" | awk '{print "Lines: " $1, "Words: " $2, "Characters: " $3}'

echo "---------------------------------"

else

echo "File $file does not exist or is not readable."

fi

Done

1. Access Logs: Record logins, logout events, and access to sensitive systems, especially noting times, locations, and user details. Track unusual login times, repeated failed access attempts, and access from unfamiliar IPs.

File Access Logs: Log read, write, and delete actions on sensitive files and directories. Monitor for bulk data access or download activity.

Authentication Logs: Record successful and failed authentication attempts for critical systems and flag patterns like password spraying or brute force attempts.

Change Management Logs: Track changes to permissions, file ownership, and configurations, which can show if someone tried to alter their own access level or change data inappropriately.

Firewalls can be leveraged to protect internal servers:

Network Segmentation: Divide the network into zones and place sensitive data in restricted, internal zones accessible only to approved IPs or subnets. Firewalls can control traffic between these zones.

Access Control Lists (ACLs): Use firewall ACLs to specify who can access internal servers by defining permitted IP addresses and protocols, blocking all others.

Application Layer Filtering: Set rules to filter traffic based on application-specific data, ensuring that only authorized users and applications can access the data on sensitive servers.

Close Unnecessary Ports: Keep only essential ports (e.g., ports for database access, web applications) open, and close all other ports to reduce the attack surface.

Port-Specific Permissions: Allow access to sensitive services only on specific ports (like port 443 for HTTPS) and restrict access to these ports to only essential IP addresses or departments.