ISSN Online : 2709-5088
ISSN Print : 2709-507X
IJCST

INTERNATIONAL JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY (IJCST) Vol. 6 No. 1 (2022)

Choosing between Kubernetes on Virtual Machines vs. Bare-Metal

Syed Afraz Ali https://orcid.org/0009-0001-6872-6786

Muhammad Waleed Zafar https://orcid.org/0009-0006-9970-6901

Abstract: The integration of OpenShift on OpenStack has emerged as a compelling solution for optimizing modern application deployment. OpenShift, a robust container orchestration platform, combines seamlessly with OpenStack's infrastructure management capabilities to create a dynamic environment that offers unparalleled scalability, flexibility, and efficiency. In this abstract, we explore the symbiotic relationship between OpenShift and OpenStack, highlighting how their combined strengths enable organizations to address diverse use cases. OpenShift's containerization and automation capabilities align seamlessly with OpenStack's resource management and networking features, resulting in a cohesive platform for deploying, scaling, and managing applications. Through a comprehensive architecture design, we examine the intricacies of deploying OpenShift on OpenStack. We delve into considerations like high availability, networking, security, and storage integration, while also addressing the nuances of CI/CD pipelines and disaster recovery strategies. The abstract further explores real-world use cases, ranging from microservices deployment to hybrid cloud scenarios, showcasing the versatility of this integration.

By understanding the convergence of OpenShift and OpenStack, organizations can unlock the potential to innovate faster, optimize resource utilization, and meet the demands of modern application deployment. This abstract provides a glimpse into the powerful synergy that these technologies bring, ultimately paving the way for enhanced application lifecycle management in today's ever-evolving technological landscape.

ISSN Online : 2709-5088
ISSN Print : 2709-507X
IJCST
INTERNATIONAL JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY (IJCST) Vol. 6 No. 1 (2022)

1. Introduction

In the dynamic realm of modern application deployment, the integration of OpenShift on OpenStack has emerged as a potent strategy to unlock unparalleled flexibility, scalability, and efficiency. OpenShift, a robust container orchestration platform, and OpenStack, a versatile infrastructure management framework, converge to provide a holistic solution that empowers organizations to streamline their application lifecycle management. OpenShift's prowess in containerization, orchestration, and automation seamlessly complements OpenStack's capabilities in resource management, networking, and security. Together, they form a symbiotic relationship that caters to a multitude of deployment scenarios and business requirements. Throughout this exploration, we will delve into the intricate details of designing architectures, understanding use cases, and crafting strategies for deploying applications on OpenShift within an OpenStack environment. By examining the synergies between these two formidable technologies, we'll uncover how organizations can harness their combined potential to navigate the complex landscape of modern application deployment with confidence and finesse.

1.1 OpenShift on OpenStack design patterns

Designing OpenShift deployments on OpenStack involves several considerations. Here are some common design patterns: Single-Node Cluster: Suitable for development or testing, where OpenShift runs on a single VM. This simplifies deployment but lacks high availability. Multi-Master Cluster: Offers high availability by having multiple master nodes. This ensures the cluster remains operational even if one master goes down. Node Scaling: Design your setup to easily scale the number of worker nodes based on demand. OpenStack's auto-scaling groups can help with this. Persistent Storage: Utilize OpenStack Cinder or Manila for persistent storage in OpenShift pods. This ensures data durability and allows stateful applications to run successfully.

Network Configuration: Set up OpenStack networking to provide connectivity between OpenShift nodes. Utilize Neutron networking features like routers, security groups, and load balancers. Load Balancing: Use OpenStack Load Balancer service or Octavia to distribute traffic across OpenShift worker nodes for enhanced performance and failover.

ISSN Online : 2709-5088
ISSN Print : 2709-507X
IJCST
INTERNATIONAL JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY (IJCST) Vol. 6 No. 1 (2022)

Security Groups and Policies: Implement proper security groups and policies in OpenStack to control network traffic between OpenShift components and nodes. External Access: Configure OpenStack Neutron routers and floating IPs to allow external access to applications deployed on OpenShift.

High Availability Patterns: Implement patterns like Active-Passive or Active-Active setups for different components (masters, etcd, worker nodes) to ensure system availability and fault tolerance.

Automated Deployment: Leverage OpenStack Heat templates or Ansible playbooks for automating the deployment and management of OpenShift clusters.

Monitoring and Logging: Integrate OpenShift monitoring tools like Prometheus and Grafana with OpenStack's logging and monitoring services for comprehensive cluster management.

Backup and Disaster Recovery: Plan for backups of both OpenShift and OpenStack components to facilitate recovery in case of data loss or system failures. Remember that each application's requirements and the existing OpenStack infrastructure can influence the design choices you make. It's essential to carefully analyze your use case and workload to create an architecture that meets your specific needs.

1.2 Components of the Architecture:

1.2.1 OpenStack Infrastructure:

Compute Nodes: VM instances that host OpenShift nodes.

Networking: Neutron for creating networks, routers, security groups.

Block Storage: Cinder for persistent storage volumes.

Load Balancing: Octavia for load balancing across OpenShift nodes.

1.2.2 OpenShift Nodes:

Master Nodes: Control plane components like API server, controller manager, scheduler, etcd.

Worker Nodes: Run application pods and containers.

ISSN Online : 2709-5088
ISSN Print : 2709-507X
IJCST
INTERNATIONAL JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY (IJCST) Vol. 6 No. 1 (2022)

1.2.3 Design Patterns and Considerations:

Multi-Master Configuration:

- Deploy at least three master nodes for high availability.
- Use etcd clustering for distributed data storage.

Worker Node Scaling:

- Leverage OpenStack's auto-scaling groups to dynamically adjust worker node count based on demand.

Networking:

- Create a dedicated OpenStack network for OpenShift communication.
- Utilize Neutron routers for routing traffic between OpenShift nodes and external networks.

Load Balancing:

- Use Octavia for load balancing incoming traffic across worker nodes.

Persistent Storage:

- Integrate OpenStack Cinder volumes for stateful application storage.
- Configure PersistentVolume and PersistentVolumeClaim resources in OpenShift.

Security Groups and Policies:

- Define OpenStack security groups and Neutron firewall rules to control communication between OpenShift nodes.

External Access:

- Assign floating IPs to OpenShift services to allow external access.
- Set up OpenStack Neutron routers to route external traffic to the appropriate OpenShift services.

High Availability Patterns:

ISSN Online : 2709-5088
ISSN Print : 2709-507X
IJCST
INTERNATIONAL JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY (IJCST) Vol. 6 No. 1 (2022)

- Deploy multiple worker nodes for application redundancy.
- Utilize Active-Passive or Active-Active configurations for key components.

Automated Deployment:

- Use OpenStack Heat templates or Ansible playbooks to automate OpenShift cluster provisioning and configuration.

Monitoring and Logging:

- Integrate OpenShift monitoring tools (Prometheus, Grafana) with OpenStack monitoring services.
- Capture OpenShift and OpenStack logs for troubleshooting and analysis.

Backup and Disaster Recovery:

- Regularly back up OpenShift etcd data and configuration.
- Backup OpenStack components and configuration using OpenStack-native tools.

Networking Flow:

- Traffic from external clients reaches OpenStack Load Balancer.
- Load Balancer routes traffic to OpenShift services running on worker nodes.
- Worker nodes communicate with OpenShift master nodes for orchestration and management.
- Communication between OpenShift master nodes is facilitated by etcd.
- OpenShift worker nodes communicate with OpenStack Cinder for persistent storage.

Remember that actual deployment details may vary based on your specific requirements, version of OpenShift, OpenStack setup, and any additional tools or integrations you choose to include. Always consult the official documentation for both OpenShift and OpenStack for the latest guidelines and best practices.

1.3 Design considerations

Resource Sizing and Scaling:

- Determine the appropriate sizing for OpenShift master and worker nodes based on workload requirements.
- Plan for horizontal scaling by utilizing OpenStack's auto-scaling capabilities.

High Availability and Redundancy:

- Design for high availability by deploying multiple master nodes and worker nodes.
- Implement redundancy for critical components to ensure system resilience.

Networking and Security:

- Configure OpenStack networks and security groups to control communication between OpenShift nodes.
- Consider using Virtual Private Networks (VPNs) or private networking for added security.

Storage Integration:

- Choose between ephemeral or persistent storage based on application needs.
- Integrate OpenStack Cinder volumes for stateful applications.
- Implement proper storage isolation and manage storage quotas.

Load Balancing and Traffic Distribution:

- Utilize OpenStack Octavia or other load balancer services to evenly distribute incoming traffic to worker nodes.
- Design load balancing for scalability and failover.

Network Latency and Performance:

- Minimize network latency by placing OpenShift nodes and OpenStack resources within the same availability zone or region.
- Leverage OpenStack's placement groups to ensure nodes are placed close to each other for improved communication.

External Access and Ingress:

- Plan for external access to OpenShift services by assigning floating IPs or setting up proper DNS records.
- Implement Ingress controllers to manage external traffic routing.

Backup and Disaster Recovery:

- Design backup and restore processes for both OpenShift and OpenStack components.
- Regularly test and validate the restoration process to ensure data recoverability.

Monitoring and Logging:

- Integrate OpenShift monitoring tools with OpenStack's monitoring services.
- Define monitoring metrics, alerts, and thresholds to proactively identify and address issues.

Upgrade and Maintenance Strategy:

- Plan for regular updates and upgrades of both OpenShift and OpenStack components.
- Have a strategy in place to minimize downtime and service disruptions during upgrades.

Automation and Orchestration:

- Leverage automation tools like Ansible, Terraform, or OpenStack Heat templates for consistent and repeatable deployment.
- Define automated workflows for provisioning, scaling, and management tasks.

Compliance and Security Policies:

- Ensure compliance with organizational security policies and industry regulations.
- Implement security best practices, including regular patching and access controls.

Integration with CI/CD Pipelines:

- Integrate OpenShift and OpenStack with continuous integration and continuous deployment (CI/CD) pipelines.
- Automate application deployment, testing, and updates.

Application and Workload Placement:

ISSN Online : 2709-5088
ISSN Print : 2709-507X
IJCST
INTERNATIONAL JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY (IJCST) Vol. 6 No. 1 (2022)

- Consider affinity and anti-affinity rules to optimize the placement of application pods on OpenShift nodes.
- Balance workloads across nodes to avoid resource bottlenecks.

Remember that successful deployment of OpenShift on OpenStack requires a deep understanding of both platforms, as well as the specific needs of your applications. Regular testing, monitoring, and adjustment of the architecture will help ensure optimal performance and reliability.

Application on boarding plan

Creating a successful application onboarding plan involves several key steps to ensure a smooth transition onto an OpenShift cluster deployed on OpenStack. Here's a comprehensive plan:

1. Assess Application Requirements:

Gather requirements for the application, including resource needs, dependencies, and performance expectations.

Determine whether the application is stateful or stateless, and whether it requires persistent storage.

2. Containerization:

Containerize the application using Docker or other compatible containerization tools.

Create a Docker file that defines the application's runtime environment and dependencies.

3. Define Deployment Strategy:

Choose between Deployment Config, Stateful Set, or other suitable resources to deploy the application on OpenShift.

Decide on the desired number of replicas for the application.

4. Application Configuration:

Define environment variables, configuration files, and other settings required for the application.

Leverage OpenShift Config Maps or Secrets for managing configuration data.

ISSN Online : 2709-5088
ISSN Print : 2709-507X
IJCST
INTERNATIONAL JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY (IJCST) Vol. 6 No. 1 (2022)

5. Persistent Storage Configuration (if applicable):

Determine whether the application requires persistent storage for data.

Set up Persistent Volume and Persistent Volume Claim resources in OpenShift.

Integrate OpenStack Cinder volumes for persistent storage.

6. Network and Ingress:

Define how the application should be accessed from outside the OpenShift cluster.

Set up Ingress resources to route external traffic to the application.

7. Testing:

Test the application within a development environment on OpenShift to ensure it functions correctly.

Verify application scaling, load balancing, and failover mechanisms.

8. Performance Optimization:

Monitor the application's performance and resource utilization.

Tune resource requests and limits to optimize application performance.

9. Backup and Recovery Testing:

Test backup and recovery procedures for the application data, as well as the application configuration on OpenShift.

10. Monitoring and Alerting:

Configure monitoring tools (e.g., Prometheus) to track application health, resource usage, and other metrics.

ISSN Online : 2709-5088
ISSN Print : 2709-507X
IJCST
INTERNATIONAL JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY (IJCST) Vol. 6 No. 1 (2022)

Set up alerts to notify administrators of any issues.

11. Documentation:

Create comprehensive documentation for deploying and maintaining the application on OpenShift.

Include step-by-step instructions, configuration details, and troubleshooting guidelines.

12. Production Deployment:

Deploy the containerized application to the production OpenShift environment.

Monitor the application closely during the initial deployment phase.

13. Load Testing and Scalability Assessment:

Conduct load testing to assess the application's scalability and performance under various conditions.

14. Rollout and Continuous Improvement:

Monitor the application's behavior in the production environment.

Continuously gather user feedback and make necessary improvements.

15. Training and Support:

Provide training to the operations team on managing and troubleshooting the application on OpenShift.

Offer support for any issues that arise during the application's lifecycle.

By following this comprehensive onboarding plan, you can ensure a successful and controlled deployment of your application on an OpenShift cluster running on the OpenStack infrastructure. Deploying applications through a CI/CD (Continuous Integration/Continuous Deployment) pipeline involves automating the steps from code development to production deployment. Here's a high-level overview of how to deploy applications on an OpenShift cluster running on OpenStack using a CI/CD pipeline:

ISSN Online : 2709-5088
ISSN Print : 2709-507X
IJCST
INTERNATIONAL JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY (IJCST) Vol. 6 No. 1 (2022)

1. Code Development:

Developers write and commit code to a version control repository (e.g., Git).

2. Continuous Integration:

A CI server (e.g., Jenkins, GitLab CI) automatically builds, tests, and packages the application code whenever changes are pushed to the repository.

Unit tests, integration tests, and other quality checks are performed during this phase.

3. Containerization:

The CI/CD pipeline uses Docker to containerize the application by building Docker images from the application code.

4. Image Registry:

The Docker images are pushed to a container image registry (e.g., Docker Hub, OpenShift Integrated Registry) for storage and distribution.

5. Continuous Deployment:

The CI/CD pipeline triggers the deployment process whenever changes are merged into a specific branch (e.g., master branch).

6. Deployment Pipeline Stages:

The deployment process can include several stages, such as development, testing, and production, each with its own OpenShift environment.

7. OpenShift Integration:

Within each deployment stage, the CI/CD pipeline interacts with the OpenShift API to deploy and manage applications.

8. Manifest Files:

ISSN Online : 2709-5088
ISSN Print : 2709-507X
IJCST
INTERNATIONAL JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY (IJCST) Vol. 6 No. 1 (2022)

Define Kubernetes manifests (YAML files) that describe how the application should be deployed in OpenShift.

Manifests include information about pods, services, routes, and any other required resources.

9. Environment Configuration:

Use OpenShift Config Maps or Secrets to inject configuration data into the application pods.

Manage environment-specific configurations for different deployment stages.

10. Deployment Strategy:

Define deployment strategies (e.g., rolling deployment) in the manifest files to control how new versions of the application are rolled out.

11. Ingress Configuration:

Set up Ingress resources to manage external access to the application within OpenShift.

12. Automation Scripts:

Use scripting tools like Bash, Python, or Ansible to automate tasks such as applying manifests, triggering deployments, and updating configurations.

13. Monitoring and Validation:

Integrate monitoring and validation steps in the CI/CD pipeline to ensure the application is functioning as expected post-deployment.

14. Version Tagging:

Use version tags on Docker images to keep track of different releases deployed to different environments.

15. Rollback Plan:

Define a rollback strategy in case issues arise after deployment. This might involve reverting to a previous known-good version.

ISSN Online : 2709-5088
ISSN Print : 2709-507X
IJCST
INTERNATIONAL JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY (IJCST) Vol. 6 No. 1 (2022)

16. Continuous Improvement:

Gather metrics, user feedback, and performance data to continuously improve the application and the deployment process.

By implementing a well-structured CI/CD pipeline, you can achieve rapid and reliable application deployments on your OpenShift cluster while maintaining consistency, automation, and traceability throughout the entire software delivery lifecycle. Certainly, here are some detailed use cases that highlight specific scenarios where deploying OpenShift on OpenStack can be advantageous:

1. Microservices Application Deployment:

Use Case: A company wants to modernize its monolithic application architecture by adopting microservices.

Solution: OpenShift on OpenStack provides a platform to deploy and manage microservices-based applications. The flexibility of OpenShift allows each microservice to run in its own container, enabling easier development, scaling, and maintenance.

2. Scalable E-commerce Platform:

Use Case: An e-commerce business experiences fluctuating traffic levels based on seasons and promotions, requiring a scalable platform.

Solution: OpenShift on OpenStack enables automatic scaling of application instances based on demand. OpenStack's auto-scaling features complement OpenShift's ability to dynamically adjust the number of replicas, ensuring optimal performance during peak periods.

3. Hybrid Cloud Deployment:

Use Case: A company wants to take advantage of both public and private clouds for different parts of their application infrastructure.

ISSN Online : 2709-5088
ISSN Print : 2709-507X
IJCST
INTERNATIONAL JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY (IJCST) Vol. 6 No. 1 (2022)

Solution: OpenShift on OpenStack allows applications to be deployed in a private cloud environment while easily integrating with public cloud services. This hybrid setup provides the flexibility to choose the best-fit environment for various components.

4. DevOps Automation:

Use Case: A development team seeks to streamline application deployment and testing processes through automation.

Solution: OpenShift on OpenStack, along with CI/CD tools, enables seamless integration of development, testing, and deployment workflows. DevOps teams can automate the building, testing, and deployment of applications, ensuring faster time to market.

5. Big Data and Analytics:

Use Case: An organization needs to process and analyze large datasets for business insights.

Solution: OpenShift on OpenStack can host big data applications such as Apache Spark or Hadoop clusters. OpenStack's resource allocation and OpenShift's container orchestration ensure efficient utilization and management of resources.

6. Stateful Applications with High Availability:

Use Case: Running stateful applications like databases or content management systems that require data persistence and high availability.

Solution: OpenShift on OpenStack can utilize OpenStack Cinder volumes to provide persistent storage for stateful applications. Multi-master configurations and node redundancy in OpenShift ensure high availability and fault tolerance.

7. Disaster Recovery Planning:

Use Case: An organization requires a disaster recovery setup for their critical applications.

Solution: OpenShift on OpenStack allows organizations to set up a disaster recovery site in a different OpenStack region or availability zone. Data replication and backup strategies can be implemented to ensure minimal data loss and downtime.

ISSN Online : 2709-5088
ISSN Print : 2709-507X
IJCST
INTERNATIONAL JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY (IJCST) Vol. 6 No. 1 (2022)

8. Edge Computing:

Use Case: Deploying applications at the network edge to reduce latency and improve performance for edge devices.

Solution: OpenShift on OpenStack enables the deployment of containerized applications at edge locations. This is especially useful for scenarios like IoT, where data processing and analysis are performed closer to the data source.

9. Compliance and Security Requirements:

Use Case: Industries with strict compliance and security regulations need a platform that offers granular control over resources and access.

Solution: OpenShift on OpenStack allows organizations to implement specific security measures, access controls, and network isolation to meet compliance requirements while benefiting from containerized application deployment.

Each of these use cases demonstrates how the combination of OpenShift's container orchestration capabilities and OpenStack's infrastructure management features can address diverse business needs and scenarios. The flexibility and scalability of this combination make it a powerful choice for a wide range of application deployment scenarios.

In conclusion, the integration of OpenShift on OpenStack presents a powerful solution for modernizing, optimizing, and streamlining application deployment and management. By combining OpenShift's container orchestration capabilities with OpenStack's infrastructure management features, organizations can achieve a dynamic and scalable platform that caters to various use cases and scenarios.

OpenShift on OpenStack offers benefits such as:

Containerization: Utilizing containers allows for consistent application packaging, portability, and efficient resource utilization.

Scalability: OpenStack's auto-scaling and OpenShift's dynamic scaling ensure applications can handle varying workloads seamlessly.

High Availability: Multi-master setups, node redundancy, and persistent storage solutions contribute to a highly available environment.

Hybrid Cloud Flexibility: The ability to deploy applications across both private and public cloud environments offers flexibility and resource optimization.

Automation: CI/CD pipelines can automate application deployment, testing, and updates, reducing manual intervention and accelerating time to market.

Resource Management: OpenStack's infrastructure management and OpenShift's resource allocation enable efficient use of computing resources.

Security and Compliance: Granular security controls and compliance measures can be implemented to protect sensitive data and adhere to regulations.

By carefully designing the architecture, considering various use cases, and implementing best practices, organizations can harness the benefits of OpenShift on OpenStack to drive innovation, enhance efficiency, and meet the demands of modern application deployment in an ever-evolving technological landscape. As you embark on your journey to deploy OpenShift on OpenStack, keep in mind the unique requirements of your organization and applications to ensure a successful and optimized deployment process.

References

[1] Mungoli, N. (2023). Scalable, Distributed AI Frameworks: Leveraging Cloud Computing for Enhanced Deep Learning Performance and Efficiency. *arXiv preprint arXiv:2304.13738*.

[2] Mughal, A. A. (2019). Cybersecurity Hygiene in the Era of Internet of Things (IoT): Best Practices and Challenges. *Applied Research in Artificial Intelligence and Cloud Computing*, *2*(1), 1-31.

[3] Mughal, A. A. (2020). Cyber Attacks on OSI Layers: Understanding the Threat Landscape. *Journal of Humanities and Applied Science Research*, *3*(1), 1-18.

[4] Mughal, A. A. (2022). Building and Securing the Modern Security Operations Center (SOC). *International Journal of Business Intelligence and Big Data Analytics*, *5*(1), 1-15.

[5] Mughal, A. A. (2019). A COMPREHENSIVE STUDY OF PRACTICAL TECHNIQUES AND METHODOLOGIES IN INCIDENT-BASED APPROACHES FOR CYBER FORENSICS. *Tensorgate Journal of Sustainable Technology and Infrastructure for Developing Countries*, *2*(1), 1-18.

[6] Mughal, A. A. (2018). The Art of Cybersecurity: Defense in Depth Strategy for Robust Protection. *International Journal of Intelligent Automation and Computing*, *1*(1), 1-20.

[7] Mughal, A. A. (2018). Artificial Intelligence in Information Security: Exploring the Advantages, Challenges, and Future Directions. *Journal of Artificial Intelligence and Machine Learning in Management*, *2*(1), 22-34.

[8] Mughal, A. A. (2022). Well-Architected Wireless Network Security. *Journal of Humanities and Applied Science Research*, *5*(1), 32-42.

[9] Mughal, A. A. (2021). Cybersecurity Architecture for the Cloud: Protecting Network in a Virtual Environment. *International Journal of Intelligent Automation and Computing*, *4*(1), 35-48.

[10] Sisodia, S., & Rocque, S. R. (2023). Underpinnings of gender bias within the context of work-life balance.

[11] Yang, L., Wang, R., Zhou, Y., Liang, J., Zhao, K., & Burleigh, S. C. (2022). An Analytical Framework for Disruption of Licklider Transmission Protocol in Mars Communications. IEEE Transactions on Vehicular Technology, 71(5), 5430-5444.

[12] Yang, L., Wang, R., Liu, X., Zhou, Y., Liu, L., Liang, J., ... & Zhao, K. (2021). Resource Consumption of a Hybrid Bundle Retransmission Approach on Deep-Space Communication Channels. *IEEE Aerospace and Electronic Systems Magazine*, *36*(11), 34-43.

[13] Liang, J., Wang, R., Liu, X., Yang, L., Zhou, Y., Cao, B., & Zhao, K. (2021, July). Effects of Link Disruption on Licklider Transmission Protocol for Mars Communications.

In *International Conference on Wireless and Satellite Systems* (pp. 98-108). Cham: Springer International Publishing.

[14] Liang, J., Liu, X., Wang, R., Yang, L., Li, X., Tang, C., & Zhao, K. (2023). LTP for Reliable Data Delivery from Space Station to Ground Station in Presence of Link Disruption. *IEEE Aerospace and Electronic Systems Magazine*.

[15] Yang, L., Liang, J., Wang, R., Liu, X., De Sanctis, M., Burleigh, S. C., & Zhao, K. (2023). A Study of Licklider Transmission Protocol in Deep-Space Communications in Presence of Link Disruptions. *IEEE Transactions on Aerospace and Electronic Systems*.

[16] Yang, L., Wang, R., Liang, J., Zhou, Y., Zhao, K., & Liu, X. (2022). Acknowledgment Mechanisms for Reliable File Transfer Over Highly Asymmetric Deep-Space Channels. *IEEE Aerospace and Electronic Systems Magazine*, *37*(9), 42-51.

[17] Zhou, Y., Wang, R., Yang, L., Liang, J., Burleigh, S. C., & Zhao, K. (2022). A Study of Transmission Overhead of a Hybrid Bundle Retransmission Approach for Deep-Space Communications. *IEEE Transactions on Aerospace and Electronic Systems*, *58*(5), 3824-3839.

[18] Yang, L., Wang, R., Liu, X., Zhou, Y., Liang, J., & Zhao, K. (2021, July). An Experimental Analysis of Checkpoint Timer of Licklider Transmission Protocol for Deep-Space Communications. In *2021 IEEE 8th International Conference on Space Mission Challenges for Information Technology (SMC-IT)* (pp. 100-106). IEEE.

[19] Zhou, Y., Wang, R., Liu, X., Yang, L., Liang, J., & Zhao, K. (2021, July). Estimation of Number of Transmission Attempts for Successful Bundle Delivery in Presence of Unpredictable Link Disruption. In *2021 IEEE 8th International Conference on Space Mission Challenges for Information Technology (SMC-IT)* (pp. 93-99). IEEE.

[20] Liang, J. (2023). *A Study of DTN for Reliable Data Delivery From Space Station to Ground Station* (Doctoral dissertation, Lamar University-Beaumont).

[21] Hussein, H. A., & Razzaq, Z. (2017). CFRP Retrofitting Schemes for Prestressed Concrete Box Beamsfor Highway Bridges. *Global Journal of Research In Engineering*.

[22] Hussein, H. A., & Razzaq, Z. (2017). Prestressed Concrete Inverted Tee Beams With CFRP for Building Structures. *Global Journal of Research In Engineering*.

[23] Hussein, H. A. (2014). *Effective CFRP Retrofitting Schemes for Prestressed Concrete Beams* (Doctoral dissertation, Old Dominion University).

[24] Hussein, H. A. (2022). *Effectiveness of Suspended Lead Dampers in Steel Buildings Under Localized Lateral Impact and Vertical Pulsating Load* (Doctoral dissertation, Old Dominion University).

[25] Hussein, H. A., & Razzaq, Z. (2021). Strengthening Prestressed Concrete Bridge Girders and Building Beams with Carbon Fiber Reinforced Polymer Sheets. *European Journal of Engineering and Technology Research*, *6*(1), 55-57.

[26] Nazarian, A., Velayati, R., Foroudi, P., Edirisinghe, D., & Atkinson, P. (2021). Organizational justice in the hotel industry: revisiting GLOBE from a national culture perspective. *International Journal of Contemporary Hospitality Management*, *33*(12), 4418-4438.

[27] Nazarian, A., Zaeri, E., Foroudi, P., Afrouzi, A. R., & Atkinson, P. (2022). Cultural perceptions of ethical leadership and its effect on intention to leave in the independent hotel industry. *International Journal of Contemporary Hospitality Management*, *34*(1), 430-455.

[28] Seyyedamiri, N., Pour, A. H., Zaeri, E., & Nazarian, A. (2022). Understanding destination brand love using machine learning and content analysis method. *Current Issues in Tourism*, *25*(9), 1451-1466.

[29] Nazarian, A., Atkinson, P., Foroudi, P., & Edirisinghe, D. (2021). Factors affecting organizational effectiveness in independent hotels–The case of Iran. *Journal of Hospitality and Tourism Management*, *46*, 293-303.

[30] Nazarian, A., & Atkinson, P. (2013). Impact of culture on leadership style: The case of Iranian organizations. *World Applied Sciences Journal*, *28*(6), 770-777.

[31] Chaudhary, J. K., Sharma, H., Tadiboina, S. N., Singh, R., Khan, M. S., & Garg, A. (2023, March). Applications of Machine Learning in Viral Disease Diagnosis. In *2023 10th International Conference on Computing for Sustainable Global Development (INDIACom)* (pp. 1167-1172). IEEE.

[32] Khan, M. S., & Minhaj, S. A. (2021). Numerical Analysis Of De Laval Nozzle Under Surrounding Zone and Compressed Flow. *International Journal for Research in Applied Science and Engineering Technology*, *9*(1), 98-105.

[33] Manikandan, N., Tadiboina, S. N., Khan, M. S., Singh, R., & Gupta, K. K. (2023, May). Automation of Smart Home for the Wellbeing of Elders Using Empirical Big Data Analysis. In *2023 3rd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)* (pp. 1164-1168). IEEE.

[34] Nallamothu, P. T., & Khan, M. S. (2023). Machine Learning for SPAM Detection. *Asian Journal of Advances in Research*, 167-179.

[35] Latha, K. H., Khan, K. A., Minhaj, S. A., & Khan, M. S. Design and Fatigue Analysis of Shot Peened Leaf Spring.

[36] Khan, M. S., & Minhaj, S. A. Design and CFD Analysis of Surgical Instrument.

[37] Khan, M. S. Control of Autonomous License Plate Recognition Drone in GPS Denied Parking Lot.

[38] Harris, H. (2023). Betting Bonanza: Strategies for Winning Big in Gambling. 1. 1-13. 10.5281/zenodo. 8106008. *Harris, Huxley.(2023). The Gambler's Grind: Secrets of Success in the Casino*, *1*, 1-14.

[39] Harris, H. (2022). Jackpot Junction: Tales of Luck and Skill in Gambling. *PAKISTAN JOURNAL OF LINGUISTICS*, *4*(4), 39-54.

[40] Harris, H. (2022). Fortune's Roll: A High-Stakes Gambling Adventure. *JOURNAL OF APPLIED LINGUISTICS AND TESOL*, *5*(4), 50-72.

[41] Haris, H. (2023). The Gambler's Grind: Secrets of Success in the Casino. *INTERNATIONAL JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY*, *7*(1), 37-52.

[42] Harris, Huxley. (2023). Casino Chronicles: Thrilling Tales from the World of Gambling. 1. 1-12. 10.5281/zenodo.8106020.

[43] Nicoli, N. (2013). Social television, creative collaboration and television production: The case of the BBC's 'the virtual revolution'. *Handbook of Social Media Management: Value Chain and Business Models in Changing Media Markets*, 603-618.

[44] Iosifidis, P., & Nicoli, N. (2020). The battle to end fake news: A qualitative content analysis of Facebook announcements on how it combats disinformation. *International Communication Gazette*, *82*(1), 60-81.

[45] Nicoli, N., & Papadopoulou, E. (2017). TripAdvisor and reputation: a case study of the hotel industry in Cyprus. *EuroMed Journal of Business*, *12*(3), 316-334.

[46] Iosifidis, P., & Nicoli, N. (2020). *Digital democracy, social media and disinformation*. Routledge.

[47] Nicoli, N. (2008). Digital television in Cyprus. *Digital Television in Europe, VUBPress*, 33-42.

[48] Nicoli, N. (2011). Creative Management, Technology and the BBC. In *Technology for Creativity and Innovation: Tools, Techniques and Applications* (pp. 285-301). IGI Global.

[49] Nicoli, N. (2012). BBC in-house production and the role of the window of creative competition. *Journal of Media Business Studies*, *9*(4), 1-19.

[50] Nicoli, N. (2014). From digital switchover to austerity measures: A case study of the Cypriot television landscape. *International Journal of Digital Television*, *5*(3), 207-220.

[51] John Samuel, I., Salem, O., & He, S. (2022). Defect-oriented supportive bridge inspection system featuring building information modeling and augmented reality. *Innovative Infrastructure Solutions*, *7*(4), 247.

[52] Salem, O., Samuel, I. J., & He, S. (2020). Bim And Vr/Ar Technologies: From Project Development To Lifecycle Asset Management. *Proceedings of the International Structural Engineering and Construction, Angamaly, India*, 14-15.

[53] Johnson, J. I., & Arulselvan, S. (2015). A Study on Relationship between Safety and Quality Performance in Foundation. *International Journal of Advance Research In Science and Engineering*, *4*(2), 218-229.

[54] Johnsamuel, I. (2012). Precast Design.

[55] Samuel, I. J., Hesarkuchak, M. T., & Salem, O. (2019, December). Multi-criteria-based simulation model to estimate resources for bridge inspections. In *2019 Winter Simulation Conference (WSC)* (pp. 3001-3007). IEEE.

[56] Samuel, I. J. (2023). *A Human-Centered Infrastructure Asset Management Framework Using BIM and Augmented Reality* (Doctoral dissertation, George Mason University).

[57] SIVAKUMAR, R., & JOHNSON, I. (2013). STUDY OF GEO-POLYMER CONCRETE WITH LIGHT WEIGHT AGGREGATES.

[58] Johnson, I., & BE, F. Y. GROUND GRANULATED BLAST FURNACE SLAG BASED LIGHT WEIGHT GEO-POLYMER CONCRETE.

[59] Mahmood, T., Fulmer, W., Mungoli, N., Huang, J., & Lu, A. (2019, October). Improving information sharing and collaborative analysis for remote geospatial visualization using mixed reality. In *2019 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)* (pp. 236-247). IEEE.

[60] Mungoli, N. (2020). *Exploring the Technological Benefits of VR in Physical Fitness* (Doctoral dissertation, The University of North Carolina at Charlotte).

[61] Mungoli, N. (2023). Adaptive Ensemble Learning: Boosting Model Performance through Intelligent Feature Fusion in Deep Neural Networks. *arXiv preprint arXiv:2304.02653*.

[62] Mungoli, N. (2023). Deciphering the Blockchain: A Comprehensive Analysis of Bitcoin's Evolution, Adoption, and Future Implications. *arXiv preprint arXiv:2304.02655*.

[63] Mungoli, N. (2023). Adaptive Feature Fusion: Enhancing Generalization in Deep Learning Models. *arXiv preprint arXiv:2304.03290*.

[64] Mungoli, N. Revolutionizing Industries: The Impact of Artificial Intelligence Technologies.

[65] Mungoli, N. Intelligent Machines: Exploring the Advancements in Artificial Intelligence.

[66] Mungoli, N. Exploring the Ethical Implications of AI-powered Surveillance Systems.

[67] Mungoli, N. Exploring the Boundaries of Artificial Intelligence: Advances and Challenges.

[68] M. Shamil, M., M. Shaikh, J., Ho, P. L., & Krishnan, A. (2014). The influence of board characteristics on sustainability reporting: Empirical evidence from Sri Lankan firms. *Asian Review of Accounting*, *22*(2), 78-97.

[69] Shaikh, I. M., Qureshi, M. A., Noordin, K., Shaikh, J. M., Khan, A., & Shahbaz, M. S. (2020). Acceptance of Islamic financial technology (FinTech) banking services by Malaysian users: an extension of technology acceptance model. *foresight*, *22*(3), 367-383.

[70] Muniapan, B., & Shaikh, J. M. (2007). Lessons in corporate governance from Kautilya's Arthashastra in ancient India. *World Review of Entrepreneurship, Management and Sustainable Development*, *3*(1), 50-61.

[71] Bhasin, M. L., & Shaikh, J. M. (2013). Voluntary corporate governance disclosures in the annual reports: an empirical study. *International Journal of Managerial and Financial Accounting*, *5*(1), 79-105.

[72] Karim, A. M., Shaikh, J. M., & Hock, O. Y. (2014). Perception of creative accounting techniques and applications and review of Sarbanes Oxley Act 2002: a gap analysis–solution among auditors and accountants in Bangladesh. *Port City International University Journal*, *1*(2), 1-12.

[73] Mamun, M. A., Shaikh, J. M., & Easmin, R. (2017). Corporate social responsibility disclosure in Malaysian business. *Academy of Strategic Management Journal*, *16*(2), 29-47.

[74] Khadaroo, I., & Shaikh, J. M. (2007). Corporate governance reforms in Malaysia: insights from institutional theory. *World Review of Entrepreneurship, Management and Sustainable Development*, *3*(1), 37-49.

[75] Bhasin, M. L., & Shaikh, J. M. (2013). Economic value added and shareholders' wealth creation: the portrait of a developing Asian country. *International Journal of Managerial and Financial Accounting*, *5*(2), 107-137.

[76] Bhasin, M. L., & Shaikh, J. M. (2013). Economic value added and shareholders' wealth creation: the portrait of a developing Asian country. *International Journal of Managerial and Financial Accounting*, *5*(2), 107-137.

[77] Asif, M. K., Junaid, M. S., Hock, O. Y., & Md Rafiqul, I. (2016). Solution of adapting creative accounting practices: an in-depth perception gap analysis among accountants and auditors of listed companies. *Australian Academy of Accounting and Finance Review*, *2*(2), 166-188.

ISSN Online : 2709-5088
ISSN Print : 2709-507X
IJCST