Hana-X Technical Landscape Infrastructure Validation Plan

1. Executive Summary

This Infrastructure Validation Plan provides a structured methodology for validating the Hana-X technical landscape, consisting of eight dedicated servers that form the backbone of our AI-driven data processing and application environment. The plan ensures that all infrastructure components meet operational requirements before deployment to production.

The validation process covers server configurations, service operations, network connectivity, security posture, and performance benchmarks. Each validation test includes clear pass/fail criteria and troubleshooting guidance for common issues. This comprehensive approach minimizes operational risks, ensures system reliability, and establishes a baseline for future infrastructure changes.

The plan is designed to be executed sequentially, starting with individual server validation, followed by service-level testing, network connectivity verification, security assessment, and performance evaluation. A final validation checklist and reporting template are provided to document the results and track remediation efforts.

2. Validation Methodology and Approach

2.1 Validation Principles

- **Comprehensive Coverage**: Test all infrastructure components, including hardware, operating systems, services, network, security, and performance.
- **Clear Pass/Fail Criteria**: Define objective metrics for each test to eliminate ambiguity.
- **Reproducibility**: Ensure tests can be repeated consistently with the same results.
- **Documentation**: Record all test results, issues encountered, and remediation actions.
- **Incremental Testing**: Start with basic components and progressively test more complex systems.

2.2 Validation Process

1. **Preparation Phase**
   - Review server specifications and configurations
   - Ensure access credentials are available
   - Prepare testing tools and scripts
   - Schedule validation activities

2. **Execution Phase**
   - Perform server-level validation
   - Validate service operations
   - Test network connectivity
   - Assess security controls
   - Evaluate performance metrics

3. **Reporting Phase**
   - Document test results
   - Identify and categorize issues
   - Develop remediation plans
   - Obtain stakeholder sign-off

2.3 Validation Tools

- **System Monitoring**: htop, top, vmstat, iostat
- **Network Testing**: ping, traceroute, iperf3, netcat, nmap
- **Service Validation**: curl, wget, Docker commands, service-specific CLI tools
- **Security Assessment**: nmap, OpenVAS, CIS-CAT, lynis
- **Performance Testing**: sysbench, stress-ng, Apache JMeter

3. Server-by-Server Validation Tests

3.1 General Server Validation (Apply to All Servers)

Test ID	Test Description	Command/ Method	Pass Criteria	Fail Criteria
SRV-001	OS Version Verification	`cat /etc/os-release`	Matches expected version	Version mismatch
SRV-002	Disk Space Availability	`df -h`	≥ 20% free space on all partitions	< 20% free space on any partition
SRV-003	Memory Availability	`free -m`	≥ 25% free memory	< 25% free memory
SRV-004	CPU Load	`uptime`	Load average < 0.7 × core count	Load average ≥ 0.7 × core count
SRV-005	System Time Sync	`timedatectl status`	NTP synchronized	NTP not synchronized
SRV-006	Critical Services Status	`systemctl list-units --state=failed`	No failed services	One or more failed services
SRV-007	Kernel Parameters	`sysctl -a | grep <key-params>`	Values match expected configuration	Values don’t match expected configuration
SRV-008	System Logs Check	`journalctl -p err -b`	No critical errors	Critical errors present

3.2 LLM Server (192.168.10.13)

Test ID	Test Description	Command/ Method	Pass Criteria	Fail Criteria
LLM-001	Ollama Service Status	`systemctl status ollama`	Active (running)	Inactive or failed
LLM-002	Ollama API Accessibility	`curl -s http://localhost:11434/api/tags`	Returns JSON with models	Connection refused or error
LLM-003	Required Models Availability	`ollama list`	All required models present	Missing models
LLM-004	GPU Availability	`nvidia-smi`	GPU detected with drivers	GPU not detected or driver issues
LLM-005	CUDA Installation	`nvcc --version`	CUDA installed (version ≥ 11.8)	CUDA not installed or version < 11.8
LLM-006	Model Inference Test	`curl -X POST http://localhost:11434/api/generate -d '{"model":"<model>","prompt":"test"}'`	Valid response with generated text	Error or timeout

3.3 Vector Database Server (192.168.10.24)

Test ID	Test Description	Command/ Method	Pass Criteria	Fail Criteria
VDB-001	Docker Status	`systemctl status docker`	Active (running)	Inactive or failed
VDB-002	Qdrant Container Status	`docker ps -f name=qdrant`	Container running	Container not running
VDB-003	Qdrant API Accessibility	`curl -s http://localhost:6333/collections`	Returns JSON response	Connection refused or error
VDB-004	Qdrant Health Check	`curl -s http://localhost:6333/health`	Returns	Error or different response
VDB-005	Qdrant Storage Persistence	Check volume mounts with `docker inspect qdrant`	Volume properly mounted	Volume not mounted
VDB-006	Qdrant Collections Check	`curl -s http://localhost:6333/collections`	Expected collections present	Missing collections

3.4 Orchestration Server (192.168.10.15)

Test ID	Test Description	Command/ Method	Pass Criteria	Fail Criteria
ORC-001	n8n Service Status	`systemctl status n8n`	Active (running)	Inactive or failed
ORC-002	n8n Web Interface	`curl -I http://localhost:5678`	HTTP 200 OK	Connection refused or error
ORC-003	n8n Database Connection	Check logs with `journalctl -u n8n`	No database connection errors	Database connection errors
ORC-004	n8n Workflows Status	API check or UI verification	Critical workflows active	Critical workflows inactive
ORC-005	n8n Credentials Encryption	Check configuration	Encryption enabled	Encryption disabled
ORC-006	n8n Queue Processing	Check execution history	Recent executions successful	Failed executions

3.5 Database Server (192.168.10.16)

Test ID	Test Description	Command/ Method	Pass Criteria	Fail Criteria
DB-001	Docker Status	`systemctl status docker`	Active (running)	Inactive or failed
DB-002	Supabase Container Status	`docker ps -f name=supabase`	All containers running	One or more containers not running
DB-003	PostgreSQL Accessibility	`docker exec supabase-db psql -U postgres -c "SELECT 1"`	Returns “1”	Connection error
DB-004	Database Backup Configuration	Check backup scripts/configuration	Backups configured and recent	Backups not configured or outdated
DB-005	PostgreSQL Logs Check	`docker logs supabase-db --tail 100`	No critical errors	Critical errors present
DB-006	Database Size Check	SQL query for database sizes	Within expected range	Exceeds expected size
DB-007	Connection Pool Status	Check configuration and active connections	Within limits	Exceeding limits

3.6 Development Server (192.168.10.17)

Test ID	Test Description	Command/ Method	Pass Criteria	Fail Criteria
DEV-001	Development Tools Installation	Check for required tools	All tools installed	Missing tools
DEV-002	Code Repository Access	`git ls-remote <repo-url>`	Repository accessible	Repository inaccessible
DEV-003	Build Environment	Test build of sample application	Builds successfully	Build fails
DEV-004	Unit Test Environment	Run test suite	Tests pass	Tests fail
DEV-005	Development Database	Check local database	Accessible and configured	Inaccessible or misconfigured
DEV-006	IDE Configuration	Check IDE installations	Properly configured	Misconfigured

3.7 Test Server (192.168.10.20)

Test ID	Test Description	Command/ Method	Pass Criteria	Fail Criteria
TST-001	Test Environment Setup	Check environment variables	Correctly configured	Misconfigured
TST-002	Test Database	Check test database	Accessible and seeded	Inaccessible or not seeded
TST-003	Test Frameworks	Check installation of test frameworks	All frameworks installed	Missing frameworks
TST-004	Integration Test Suite	Run integration tests	Tests pass	Tests fail
TST-005	Load Testing Tools	Check JMeter/ Locust installation	Properly installed	Not installed or misconfigured
TST-006	Test Reporting	Check report generation	Reports generated correctly	Reports not generated

3.8 DevOps Server (192.168.10.18)

Test ID	Test Description	Command/ Method	Pass Criteria	Fail Criteria
DOP-001	CI/CD Tools Status	Check Jenkins/ GitLab CI status	Services running	Services not running
DOP-002	Pipeline Configurations	Validate pipeline definitions	Valid configurations	Invalid configurations
DOP-003	Artifact Repository	Check Nexus/Artifactory status	Service running and accessible	Service down or inaccessible
DOP-004	Infrastructure as Code	Validate Terraform/Ansible files	Files validate	Validation errors
DOP-005	Monitoring Tools	Check Prometheus/Grafana status	Services running	Services not running
DOP-006	Logging Infrastructure	Check ELK/Loki status	Services running	Services not running

3.9 DevOps Workstation (192.168.10.19)

Test ID	Test Description	Command/ Method	Pass Criteria	Fail Criteria
WKS-001	WSL2 Status	Check WSL status	Running properly	Not running or errors
WKS-002	Development Tools	Check required tools in WSL	All tools installed	Missing tools
WKS-003	Remote Access Configuration	Check SSH configurations	Properly configured	Misconfigured
WKS-004	Docker Desktop	Check Docker Desktop status	Running properly	Not running or errors
WKS-005	GUI Tools	Check required GUI tools	All tools installed	Missing tools
WKS-006	Remote Repository Access	Test repository access	Repositories accessible	Repositories inaccessible

4. Service-Level Validation Tests

4.1 Ollama Service Validation

Test ID	Test Description	Command/Method	Pass Criteria	Fail Criteria
SVC-LLM-001	Ollama API Health	`curl http://192.168.10.13:11434/api/health`	Returns healthy status	Error or timeout
SVC-LLM-002	Model Loading Time	Time to load a model	< 30 seconds	≥ 30 seconds
SVC-LLM-003	Inference Performance	Measure tokens/second	≥ 10 tokens/second	< 10 tokens/second
SVC-LLM-004	Concurrent Requests	Test multiple simultaneous requests	All requests processed	Requests timeout or fail
SVC-LLM-005	Model Management	Test model pull/remove operations	Operations complete successfully	Operations fail

4.2 Qdrant Service Validation

Test ID	Test Description	Command/ Method	Pass Criteria	Fail Criteria
SVC-VDB-001	Qdrant REST API	`curl http://192.168.10.24:6333/collections`	Returns collections list	Error or timeout
SVC-VDB-002	Vector Search Performance	Test search latency	< 100ms for 1000 vectors	≥ 100ms for 1000 vectors
SVC-VDB-003	Collection Creation	Create test collection	Collection created successfully	Creation fails
SVC-VDB-004	Vector Insertion	Insert test vectors	Vectors inserted successfully	Insertion fails
SVC-VDB-005	Persistence After Restart	Restart container and check data	Data persists	Data lost

4.3 n8n Service Validation

Test ID	Test Description	Command/ Method	Pass Criteria	Fail Criteria
SVC-ORC-001	n8n Web Interface	`curl http://192.168.10.15:5678`	Returns web interface	Error or timeout
SVC-ORC-002	Workflow Execution	Execute test workflow	Workflow completes successfully	Workflow fails
SVC-ORC-003	External Service Integration	Test integration with external service	Integration works	Integration fails
SVC-ORC-004	Webhook Functionality	Test webhook trigger	Webhook triggers workflow	Webhook fails
SVC-ORC-005	Scheduled Workflows	Check scheduled workflow execution	Executes on schedule	Doesn’t execute on schedule

4.4 Supabase/PostgreSQL Service Validation

Test ID	Test Description	Command/ Method	Pass Criteria	Fail Criteria
SVC-DB-001	PostgreSQL Connection	Connect with psql client	Connection established	Connection fails
SVC-DB-002	Database CRUD Operations	Test insert/select/update/delete	Operations complete successfully	Operations fail
SVC-DB-003	Database Backup/Restore	Test backup and restore	Backup/restore successful	Backup/restore fails
SVC-DB-004	Connection Pooling	Test with multiple connections	Connections handled properly	Connection errors
SVC-DB-005	Supabase API	Test REST API endpoints	API responds correctly	API errors

4.5 CI/CD Service Validation

Test ID	Test Description	Command/ Method	Pass Criteria	Fail Criteria
SVC-CICD-001	Pipeline Execution	Trigger test pipeline	Pipeline completes successfully	Pipeline fails
SVC-CICD-002	Artifact Generation	Check artifact creation	Artifacts created correctly	Artifacts not created
SVC-CICD-003	Deployment Process	Test deployment to test environment	Deployment succeeds	Deployment fails
SVC-CICD-004	Rollback Procedure	Test rollback functionality	Rollback succeeds	Rollback fails
SVC-CICD-005	Notification System	Test pipeline notifications	Notifications sent correctly	Notifications not sent

5. Network Connectivity Validation Tests

5.1 Basic Connectivity Tests

Test ID	Test Description	Command/ Method	Pass Criteria	Fail Criteria
NET-001	Gateway Connectivity	`ping -c 4 192.168.10.1`	All packets received	Packet loss
NET-002	DNS Resolution	`nslookup google.com`	Resolves correctly	Resolution fails
NET-003	Internet Connectivity	`curl -I https://www.google.com`	HTTP 200 OK	Connection fails
NET-004	Internal Network Latency	`ping -c 20 <server-ip>`	Average < 2ms	Average ≥ 2ms
NET-005	Traceroute to Gateway	`traceroute 192.168.10.1`	Direct route with 1 hop	Multiple hops or failures

5.2 Server-to-Server Connectivity Matrix

Source Server	Destination Server	Port(s)	Test Command	Pass Criteria	Fail Criteria
LLM Server	Vector DB Server	6333	`nc -zv 192.168.10.24 6333`	Connection succeeds	Connection fails
LLM Server	Orchestration Server	5678	`nc -zv 192.168.10.15 5678`	Connection succeeds	Connection fails
Vector DB Server	LLM Server	11434	`nc -zv 192.168.10.13 11434`	Connection succeeds	Connection fails
Vector DB Server	Database Server	5432	`nc -zv 192.168.10.16 5432`	Connection succeeds	Connection fails
Orchestration Server	LLM Server	11434	`nc -zv 192.168.10.13 11434`	Connection succeeds	Connection fails
Orchestration Server	Vector DB Server	6333	`nc -zv 192.168.10.24 6333`	Connection succeeds	Connection fails
Orchestration Server	Database Server	5432	`nc -zv 192.168.10.16 5432`	Connection succeeds	Connection fails
Development Server	All Servers	Various	`for ip in 13 15 16 18 19 20 24; do nc -zv 192.168.10.$ip <port>; done`	All connections succeed	Any connection fails
Test Server	All Servers	Various	`for ip in 13 15 16 17 18 19 24; do nc -zv 192.168.10.`	All connections succeed	Any connection fails

Source Server	Destination Server	Port(s)	Test Command	Pass Criteria	Fail Criteria
			`$ip <port>; done`		
DevOps Server	All Servers	Various	`for ip in 13 15 16 17 19 20 24; do nc -zv 192.168.10.$ip <port>; done`	All connections succeed	Any connection fails
DevOps Workstation	All Servers	Various	Test with PowerShell or WSL	All connections succeed	Any connection fails

5.3 Bandwidth and Throughput Tests

Test ID	Test Description	Command/ Method	Pass Criteria	Fail Criteria
NET-BW-001	LLM to Vector DB Throughput	`iperf3 -c 192.168.10.24 -t 30`	≥ 500 Mbps	< 500 Mbps
NET-BW-002	Database to App Servers Throughput	`iperf3 -c 192.168.10.16 -t 30`	≥ 500 Mbps	< 500 Mbps
NET-BW-003	Network Stability Test	`iperf3 -c <server-ip> -t 300 -i 10`	Stable throughput, < 5% variation	Unstable throughput, ≥ 5% variation
NET-BW-004	UDP Performance Test	`iperf3 -c <server-ip> -u -b 100M`	< 1% packet loss	≥ 1% packet loss
NET-BW-005	Multi-Connection Performance	`iperf3 -c <server-ip> -P 10`	Combined throughput ≥ 900 Mbps	Combined throughput < 900 Mbps

6. Security Validation Tests

6.1 Network Security Tests

Test ID	Test Description	Command/ Method	Pass Criteria	Fail Criteria
SEC-NET-001	Open Ports Scan	`nmap -sS <server-ip>`	Only required ports open	Unnecessary ports open
SEC-NET-002	Firewall Configuration	Check iptables/ ufw rules	Rules match security policy	Rules don't match security policy
SEC-NET-003	SSH Configuration	`ssh -v <server-ip>`	Password auth disabled, key auth only	Password auth enabled
SEC-NET-004	TLS/SSL Configuration	`nmap --script ssl-enum-ciphers -p 443 <server-ip>`	Strong ciphers only	Weak ciphers allowed
SEC-NET-005	Network Segmentation	Verify routing tables and firewall rules	Proper segmentation in place	Improper segmentation

6.2 System Security Tests

Test ID	Test Description	Command/ Method	Pass Criteria	Fail Criteria
SEC-SYS-001	User Account Audit	Check /etc/passwd and groups	Only required accounts exist	Unnecessary accounts exist
SEC-SYS-002	Sudo Configuration	Check /etc/sudoers	Follows principle of least privilege	Overly permissive
SEC-SYS-003	File Permissions	Check critical file permissions	Proper permissions set	Improper permissions
SEC-SYS-004	Password Policy	Check PAM configuration	Strong password policy enforced	Weak password policy
SEC-SYS-005	System Updates	`apt update && apt list --upgradable`	No security updates pending	Security updates pending

6.3 Application Security Tests

Test ID	Test Description	Command/ Method	Pass Criteria	Fail Criteria
SEC-APP-001	Service Authentication	Test service auth mechanisms	Strong auth mechanisms in place	Weak auth mechanisms
SEC-APP-002	API Security	Test API endpoints for security	Proper authentication and authorization	Missing or weak auth
SEC-APP-003	Container Security	Scan container images	No critical vulnerabilities	Critical vulnerabilities present
SEC-APP-004	Secrets Management	Check for hardcoded secrets	No hardcoded secrets	Hardcoded secrets found
SEC-APP-005	Logging and Monitoring	Check logging configuration	Comprehensive logging enabled	Insufficient logging

7. Performance Validation Tests

7.1 System Performance Tests

Test ID	Test Description	Command/ Method	Pass Criteria	Fail Criteria
PERF-SYS-001	CPU Stress Test	`stress-ng --cpu 8 --timeout 60s`	Load handled without issues	System becomes unresponsive
PERF-SYS-002	Memory Stress Test	`stress-ng --vm 2 --vm-bytes 75% --timeout 60s`	Memory pressure handled	OOM killer activated
PERF-SYS-003	Disk I/O Performance	`fio --name=test --size=1G --rw=randrw`	Read/Write > 100MB/s	Read/Write ≤ 100MB/s
PERF-SYS-004	System Load Under Stress	Monitor load during stress tests	Load < 80% of capacity	Load ≥ 80% of capacity
PERF-SYS-005	Boot Time	Measure time from power on to services available	< 2 minutes	≥ 2 minutes

7.2 Application Performance Tests

Test ID	Test Description	Command/ Method	Pass Criteria	Fail Criteria
PERF-APP-001	LLM Inference Latency	Measure response time for standard prompt	< 2 seconds	≥ 2 seconds
PERF-APP-002	Vector DB Query Performance	Measure vector search time	< 50ms	≥ 50ms
PERF-APP-003	Database Query Performance	Run benchmark queries	< 100ms for standard queries	≥ 100ms for standard queries
PERF-APP-004	n8n Workflow Execution Time	Measure workflow execution time	Within expected timeframe	Exceeds expected timeframe
PERF-APP-005	API Response Time	Measure API endpoint response times	< 200ms	≥ 200ms

7.3 Network Performance Tests

Test ID	Test Description	Command/ Method	Pass Criteria	Fail Criteria
PERF-NET-001	Network Latency	`ping -c 100 <server-ip>`	Average < 1ms	Average ≥ 1ms
PERF-NET-002	Network Throughput	`iperf3 -c <server-ip> -t 60`	> 900 Mbps	≤ 900 Mbps
PERF-NET-003	Connection Establishment Time	Measure TCP connection time	< 10ms	≥ 10ms
PERF-NET-004	Network Stability	Long-running ping test	< 0.1% packet loss	≥ 0.1% packet loss
PERF-NET-005	DNS Resolution Time	Measure DNS query time	< 20ms	≥ 20ms

8. Troubleshooting Guide for Common Validation Issues

8.1 Server-Level Issues

8.1.1 Server Unreachable

1. Verify physical network connection
2. Check IP configuration: `ip addr show`
3. Verify gateway configuration: `ip route show`
4. Check firewall rules: `sudo iptables -L` or `sudo ufw status`
5. Verify DNS configuration: `cat /etc/resolv.conf`

8.1.2 Disk Space Issues

1. Identify full partitions: `df -h`
2. Find large files/directories: `du -h --max-depth=1 /path/to/check | sort -hr`
3. Check for and remove old log files: `find /var/log -type f -name "*.gz" -delete`
4. Clear package cache: `apt clean`
5. Remove old kernels: `apt autoremove`

8.1.3 High CPU/Memory Usage

1. Identify resource-intensive processes: `top` or `htop`
2. Check system load: `uptime`
3. Review process details: `ps aux | grep <process-id>`
4. Check for memory leaks: `smem -tk`
5. Review system logs: `journalctl -p err`

8.2 Service-Level Issues

8.2.1 Docker Container Issues

1. Check container status: `docker ps -a`
2. View container logs: `docker logs <container-id>`
3. Inspect container configuration: `docker inspect <container-id>`

4. Restart container: `docker restart <container-id>`
5. Check Docker daemon status: `systemctl status docker`

8.2.2 Ollama Service Issues

1. Check service status: `systemctl status ollama`
2. Review logs: `journalctl -u ollama`
3. Verify model availability: `ollama list`
4. Check GPU availability: `nvidia-smi`
5. Restart service: `systemctl restart ollama`

8.2.3 Database Connection Issues

1. Verify PostgreSQL service: `systemctl status postgresql`
2. Check connection parameters: `psql -h <host> -p <port> -U <user> -d <database>`
3. Verify firewall rules for port 5432
4. Check PostgreSQL logs: `/var/log/postgresql/postgresql-*.log`
5. Verify client authentication configuration: `pg_hba.conf`

8.3 Network Connectivity Issues

8.3.1 Connection Timeouts

1. Check physical network connectivity
2. Verify IP address configuration
3. Test basic connectivity: `ping <destination-ip>`
4. Check for packet loss: `ping -c 100 <destination-ip>`
5. Trace route to destination: `traceroute <destination-ip>`

8.3.2 Port Connectivity Issues

1. Verify service is listening: `ss -tulpn | grep <port>`
2. Check firewall rules: `sudo iptables -L | grep <port>`
3. Test port connectivity: `nc -zv <destination-ip> <port>`
4. Verify service configuration for correct binding address
5. Check for port conflicts: `lsof -i :<port>`

8.3.3 DNS Resolution Issues

1. Test DNS resolution: `nslookup <domain>`
2. Check DNS configuration: `cat /etc/resolv.conf`
3. Verify DNS server accessibility: `ping <dns-server-ip>`
4. Try alternative DNS server: `nslookup <domain> 8.8.8.8`
5. Check local hosts file: `cat /etc/hosts`

8.4 Security Issues

8.4.1 Failed Authentication

1. Verify credentials are correct
2. Check account lockout status
3. Review authentication logs: `journalctl -u sshd`
4. Verify SSH key permissions: `chmod 600 ~/.ssh/id_rsa`
5. Check for PAM configuration issues: `/etc/pam.d/`

8.4.2 Certificate Issues

1. Verify certificate validity: `openssl x509 -in <cert-file> -text -noout`
2. Check certificate expiration: `openssl x509 -in <cert-file> -noout -enddate`
3. Verify certificate chain: `openssl verify -CAfile <ca-file> <cert-file>`
4. Check for certificate revocation
5. Verify hostname matches certificate CN/SAN

8.5 Performance Issues

8.5.1 Slow Response Times

1. Check system load: `uptime`
2. Monitor resource usage: `top` or `htop`
3. Check disk I/O: `iostat -x 1`
4. Monitor network traffic: `iftop` or `nethogs`
5. Check for slow queries in database: `pg_stat_statements`

8.5.2 Resource Contention

1. Identify CPU-bound processes: `top` (sort by CPU)
2. Identify memory-bound processes: `top` (sort by memory)
3. Check for disk contention: `iostat -x 1`
4. Monitor network saturation: `iftop`
5. Adjust resource limits or scheduling priorities

9. Validation Checklist and Reporting Template

9.1 Validation Checklist

Category	Validation Area	Status	Issues	Notes
Server	LLM Server (192.168.10.13)	□ Pass □ Fail		
Server	Vector DB Server (192.168.10.24)	□ Pass □ Fail		
Server	Orchestration Server (192.168.10.15)	□ Pass □ Fail		
Server	Database Server (192.168.10.16)	□ Pass □ Fail		
Server	Development Server (192.168.10.17)	□ Pass □ Fail		
Server	Test Server (192.168.10.20)	□ Pass □ Fail		
Server	DevOps Server (192.168.10.18)	□ Pass □ Fail		
Server	DevOps Work-station (192.168.10.19)	□ Pass □ Fail		
Service	Ollama	□ Pass □ Fail		
Service	Qdrant	□ Pass □ Fail		
Service	n8n	□ Pass □ Fail		
Service	Supabase/Post-greSQL	□ Pass □ Fail		
Service	CI/CD Pipeline	□ Pass □ Fail		
Network	Basic Connectiv-ity	□ Pass □ Fail		
Network	Server-to-Server Connectivity	□ Pass □ Fail		

Category	Validation Area	Status	Issues	Notes
Network	Bandwidth and Throughput	□ Pass □ Fail		
Security	Network Security	□ Pass □ Fail		
Security	System Security	□ Pass □ Fail		
Security	Application Security	□ Pass □ Fail		
Performance	System Performance	□ Pass □ Fail		
Performance	Application Performance	□ Pass □ Fail		
Performance	Network Performance	□ Pass □ Fail		

9.2 Issue Tracking

Issue ID	Related Test ID	Description	Severity	Status	Assigned To	Resolution
			□ Critical □ High □ Medium □ Low	□ Open □ In Progress □ Resolved		
			□ Critical □ High □ Medium □ Low	□ Open □ In Progress □ Resolved		
			□ Critical □ High □ Medium □ Low	□ Open □ In Progress □ Resolved		
			□ Critical □ High □ Medium □ Low	□ Open □ In Progress □ Resolved		
			□ Critical □ High □ Medium □ Low	□ Open □ In Progress □ Resolved		

9.3 Validation Report Summary

Validation Date: ___

Validation Performed By: ___

Overall Status: □ Pass □ Conditional Pass □ Fail

Executive Summary:

[Provide a brief summary of the validation results, highlighting major findings, issues, and recommendations]

Critical Issues:

[List any critical issues that must be addressed before production deployment]

Recommendations:

[Provide recommendations for addressing issues and improving the infrastructure]

Sign-off:

Validator: ___ ***Date:*** ___

Infrastructure Manager: ___ ***Date:*** ___

IT Security Officer: ___ ***Date:*** ___

Appendix A: Validation Scripts

```
#!/bin/bash
# server_validation.sh - Basic server validation script

echo "Running basic server validation for $(hostname)"
echo "================================================"

echo "OS Version:"
cat /etc/os-release

echo -e "\nDisk Space:"
df -h

echo -e "\nMemory Usage:"
free -m

echo -e "\nCPU Load:"
uptime

echo -e "\nFailed Services:"
systemctl list-units --state=failed

echo -e "\nSystem Errors:"
journalctl -p err -b --no-pager | tail -n 20

echo -e "\nNetwork Interfaces:"
ip addr show

echo -e "\nOpen Ports:"
ss -tulpn

echo "================================================"
echo "Validation complete"
```

Appendix B: Network Validation Matrix

```bash
#!/bin/bash
# network_validation.sh - Network connectivity validation script

SERVERS=(
  "192.168.10.13:LLM_Server:11434"
  "192.168.10.24:Vector_DB:6333"
  "192.168.10.15:Orchestration:5678"
  "192.168.10.16:Database:5432"
  "192.168.10.17:Development:22"
  "192.168.10.20:Test:22"
  "192.168.10.18:DevOps:22"
  "192.168.10.19:Workstation:22"
)

echo "Network Connectivity Matrix"
echo "==========================="
printf "%-20s" "From/To"
for server in "${SERVERS[@]}"; do
  IFS=':' read -r ip name port <<< "$server"
  printf "%-15s" "$name"
done
echo ""

for source in "${SERVERS[@]}"; do
  IFS=':' read -r src_ip src_name src_port <<< "$source"
  printf "%-20s" "$src_name"

  for target in "${SERVERS[@]}"; do
    IFS=':' read -r tgt_ip tgt_name tgt_port <<< "$target"
    if [ "$src_ip" == "$tgt_ip" ]; then
      printf "%-15s" "N/A"
    else
      timeout 1 bash -c "echo > /dev/tcp/$tgt_ip/$tgt_port" 2>/dev/null
      if [ $? -eq 0 ]; then
        printf "%-15s" "✓"
      else
        printf "%-15s" "✗"
      fi
    fi
  done
  echo ""
done
```

Appendix C: Service Validation Scripts

```
#!/bin/bash
# ollama_validation.sh - Validate Ollama service

echo "Validating Ollama Service"
echo "========================="

# Check service status
echo "Service Status:"
systemctl status ollama | grep Active

# Check available models
echo -e "\nAvailable Models:"
curl -s http://localhost:11434/api/tags | grep name

# Test basic inference
echo -e "\nBasic Inference Test:"
time curl -s -X POST http://localhost:11434/api/generate -d '{"mod-
el":"llama2","prompt":"Hello, world!","max_tokens":10}' | grep text

echo "========================="
```