NVIDIA Base Command Manager - Detailed Procedures & Solutions for NCP-AIOPS

1. Installation & Initial Configuration

Pre-Installation Requirements

bash

```
# System Requirements Check
- Head Node: 64GB RAM minimum, 1TB storage
- Compute Nodes: GPU-enabled systems with CUDA drivers
- Network: InfiniBand or high-speed Ethernet
- OS: RHEL 8.x, Ubuntu 20.04+ or SLES 15

# Network Prerequisites
- DHCP server configuration
- DNS resolution setup
- NTP synchronization
- Firewall rules for cluster communication
```

Installation Process

bash

```
# 1. Download and extract BCM installer
wget https://nvidia.com/base-command-manager-installer.tar.gz
tar -xzf base-command-manager-installer.tar.gz

# 2. Run installation wizard
./install --wizard
# Follow prompts for:
# - Network configuration
# - Storage setup
# - License activation
# - Initial admin user creation

# 3. Verify installation
cm-cluster-status
cm-node-list
```

Post-Installation Configuration

```bash
# Configure cluster settings
cm-cluster-configure --dns-domain cluster.local
cm-cluster-configure --timezone UTC

# Setup shared storage
cm-storage-configure --type nfs --path /shared

# Initialize GPU monitoring
cm-gpu-initialize --all-nodes
```

---

2. Cluster Provisioning & Node Management

Node Provisioning Process

bash

```
# 1. Add compute nodes to cluster
cm-node-add --hostname compute-01 --mac 00:11:22:33:44:55
cm-node-add --hostname compute-02 --mac 00:11:22:33:44:56

# 2. Assign software images
cm-image-assign --node compute-01 --image ubuntu-20.04-gpu
cm-image-assign --node compute-02 --image ubuntu-20.04-gpu

# 3. Boot nodes
cm-node-boot --node compute-01,compute-02

# 4. Monitor provisioning status
cm-node-status --verbose
```

Software Image Management

bash

```
# Create custom software image
cm-image-create --name "ai-workload-v1" --base ubuntu-20.04
cm-chroot ai-workload-v1

# Inside chroot environment:
apt update && apt install -y nvidia-docker2 kubernetes-node
pip install torch tensorflow rapids-cudf
exit

# Finalize and deploy image
cm-image-finalize ai-workload-v1
cm-image-deploy --image ai-workload-v1 --nodes compute-group-gpu
```

Hardware Discovery & Configuration

```
bash

# GPU discovery and configuration
cm-gpu-discover --all-nodes
cm-gpu-configure --enable-persistence-mode
cm-gpu-configure --power-limit 250W

# Network interface configuration
cm-network-configure --interface ib0 --ip-range 192.168.100.0/24
cm-network-configure --interface eth0 --dhcp

# Storage configuration
cm-storage-mount --type lustre --server storage-server:/lustre --mountpoint /scratch
```

---

3. Workload Management & Job Scheduling

Kubernetes Configuration

bash

```
# Enable Kubernetes orchestration
cm-kubernetes-enable --version 1.25
cm-gpu-operator-install --version 23.3.2

# Configure GPU scheduling
kubectl apply -f - <<EOF
apiVersion: v1
kind: ResourceQuota
metadata:
  name: gpu-quota
spec:
  hard:
    nvidia.com/gpu: "8"
    requests.memory: "64Gi"
    requests.cpu: "32"
EOF

# Deploy GPU-enabled workload
kubectl apply -f - <<EOF
apiVersion: v1
kind: Pod
metadata:
  name: gpu-pod
spec:
  containers:
  - name: gpu-container
    image: nvidia/cuda:11.8-runtime-ubuntu20.04
    resources:
      limits:
        nvidia.com/gpu: 1
```

```
    command: ["/bin/bash", "-c", "nvidia-smi && sleep 3600"]
EOF
```

Slurm Integration

```bash
# Configure Slurm workload manager
cm-slurm-configure --partition gpu --nodes compute[01-04]
cm-slurm-configure --partition cpu --nodes compute[05-08]

# Create Slurm configuration
cat > /etc/slurm/slurm.conf <<EOF
ClusterName=ai-cluster
ControlMachine=head-node
SlurmUser=slurm

# GPU Partition
PartitionName=gpu Nodes=compute[01-04] Default=YES MaxTime=24:00:00 State=UP
NodeName=compute[01-04] CPUs=32 RealMemory=128000 Gres=gpu:4 State=UNKNOWN

# CPU Partition
PartitionName=cpu Nodes=compute[05-08] MaxTime=48:00:00 State=UP
NodeName=compute[05-08] CPUs=64 RealMemory=256000 State=UNKNOWN
EOF

# Submit GPU job example
sbatch --partition=gpu --gres=gpu:2 --mem=32G gpu-training.sh
```

Job Submission Examples

```bash
```

```
# AI Training Job Script (gpu-training.sh)
#!/bin/bash
#SBATCH --job-name=ai-training
#SBATCH --partition=gpu
#SBATCH --gres=gpu:4
#SBATCH --mem=64G
#SBATCH --time=12:00:00

module load cuda/11.8
module load python/3.9

python train_model.py --gpus 4 --batch-size 128 --epochs 100

# HPC Simulation Job
#!/bin/bash
#SBATCH --job-name=hpc-sim
#SBATCH --partition=cpu
#SBATCH --ntasks=256
#SBATCH --mem-per-cpu=4G
#SBATCH --time=24:00:00

mpirun -np 256 simulation.exe --input config.xml
```

---

4. Monitoring & Performance Optimization

Real-Time Monitoring Setup

```
bash
```

```
# Enable comprehensive monitoring
cm-monitoring-enable --components dcgm,prometheus,grafana
cm-monitoring-configure --retention-days 30

# DCGM (Data Center GPU Manager) configuration
dcgmi discovery -l  # List all GPUs
dcgmi stats -g 0 -e  # Enable stats collection
dcgmi health -g 0  # Check GPU health

# Custom monitoring queries
dcgmi stats -g 0 --verbose  # Detailed GPU metrics
dcgmi dmon -e 155,150,203,204  # Monitor specific metrics
```

Performance Metrics Collection

```
bash

# System performance monitoring
cm-perf-monitor --interval 60 --metrics cpu,memory,network,gpu
cm-perf-alert --metric gpu-temp --threshold 85 --action email

# Job performance analysis
cm-job-analyze --job-id 12345 --metrics gpu-util,memory-usage
cm-job-compare --job-ids 12345,12346 --output report.html

# Network performance testing
cm-network-test --test bandwidth --nodes compute[01-04]
cm-network-test --test latency --protocol rdma
```

Performance Optimization Techniques

```bash
# GPU optimization
nvidia-smi -pm 1  # Enable persistence mode
nvidia-smi -pl 300  # Set power limit
nvidia-smi --auto-boost-default=DISABLED  # Disable auto-boost

# Memory optimization
echo 'vm.swappiness=1' >> /etc/sysctl.conf
echo 'vm.zone_reclaim_mode=1' >> /etc/sysctl.conf

# Network optimization for AI workloads
echo 'net.core.rmem_max=134217728' >> /etc/sysctl.conf
echo 'net.core.wmem_max=134217728' >> /etc/sysctl.conf
echo 'net.ipv4.tcp_congestion_control=bbr' >> /etc/sysctl.conf
```

---

5. Troubleshooting Procedures

Common Node Issues & Solutions

Node Boot Failures

```bash
```

```
# Diagnosis steps
cm-node-status compute-01 --detailed
cm-log-view --node compute-01 --service dhcp
cm-log-view --node compute-01 --service pxe

# Common fixes
# 1. MAC address mismatch
cm-node-edit compute-01 --mac-correct 00:11:22:33:44:57

# 2. Network boot issues
cm-pxe-regenerate --node compute-01
cm-dhcp-restart

# 3. Image corruption
cm-image-verify ubuntu-20.04-gpu
cm-image-rebuild ubuntu-20.04-gpu --force
```

GPU Recognition Issues

bash

```
# GPU troubleshooting
nvidia-smi  # Check if GPUs are visible
lspci | grep -i nvidia  # Verify hardware detection

# Driver issues
cm-driver-update --component nvidia --nodes compute-01
cm-node-reboot compute-01

# CUDA toolkit verification
nvcc --version
/usr/local/cuda/samples/1_Utilities/deviceQuery/deviceQuery
```

Workload Management Issues

Job Scheduling Problems

bash

```
# Slurm troubleshooting
sinfo -R  # Check node reasons
squeue --start  # Show job start times
sacct -j 12345 --format=JobID,State,ExitCode,DerivedExitCode

# Common fixes
# 1. Node drain issues
scontrol update NodeName=compute-01 State=RESUME

# 2. Resource allocation problems
scontrol show job 12345
scontrol update JobId=12345 TimeLimit=48:00:00

# 3. GPU allocation issues
scontrol show partition gpu
scontrol update PartitionName=gpu MaxTime=24:00:00
```

Kubernetes Issues

bash

```
# Pod troubleshooting
kubectl describe pod gpu-pod
kubectl logs gpu-pod
kubectl get events --sort-by='.lastTimestamp'

# GPU operator issues
kubectl get pods -n gpu-operator-resources
kubectl logs -n gpu-operator-resources gpu-operator-xxx

# Node issues
kubectl describe node compute-01
kubectl get nodes -o wide
```

Performance Troubleshooting

GPU Underutilization

bash

```
# Diagnosis
nvidia-smi dmon -s pucvmet -i 1  # Monitor GPU metrics
dcgmi dmon -e 150,155,203,204  # DCGM monitoring

# Common causes and fixes
# 1. CPU bottleneck
top -p $(pgrep python)
taskset -cp 0-31 $(pgrep python)  # Pin to specific CPUs

# 2. I/O bottleneck
iotop -p $(pgrep python)
# Move data to faster storage or increase buffer sizes

# 3. Memory bandwidth issues
numactl --hardware
numactl --cpubind=0 --membind=0 python train.py
```

Network Performance Issues

bash

```
# InfiniBand troubleshooting
ibstat  # Check IB port status
ibdiagnet  # Network topology and health check
perfquery  # Performance counters

# Ethernet troubleshooting
ethtool eth0  # Check link status and settings
iperf3 -s  # Server mode for bandwidth testing
iperf3 -c server-ip -t 60  # Client bandwidth test
```

---

6. Security & Access Management

User Authentication Setup

```bash
# LDAP integration
cm-auth-configure --type ldap --server ldap.company.com
cm-auth-configure --base-dn "dc=company,dc=com"
cm-auth-configure --bind-user "cn=admin,dc=company,dc=com"

# OIDC configuration
cm-auth-configure --type oidc --provider https://auth.company.com
cm-auth-configure --client-id cluster-auth --client-secret xxx

# Local user management
cm-user-add --username scientist1 --groups gpu-users,data-scientists
cm-user-quota --username scientist1 --gpu-hours 100 --storage 1TB
```

Role-Based Access Control

```bash
```

```
# Create custom roles
cm-role-create --name "ml-engineer" --permissions job-submit,job-monitor
cm-role-create --name "admin" --permissions cluster-admin,user-admin

# Assign roles to users
cm-user-role-assign --username scientist1 --role ml-engineer
cm-group-role-assign --group data-team --role ml-engineer

# Resource access control
cm-partition-access --partition gpu --groups gpu-users
cm-storage-access --path /datasets --groups data-scientists --mode ro
```

Security Hardening

```bash
# Network security
cm-firewall-enable --strict
cm-firewall-rule --service ssh --port 22 --source admin-network
cm-firewall-rule --service slurm --port 6817-6818 --source cluster-network

# Certificate management
cm-cert-generate --ca cluster-ca --validity 365
cm-cert-deploy --service kubernetes --cert cluster-cert
cm-cert-renew --auto-enable --days-before 30

# Audit logging
cm-audit-enable --events user-login,job-submit,admin-actions
cm-audit-configure --retention 90 --export syslog
```

---

7. Cloud & Hybrid Integration

Cloud Burst Configuration

```bash
# AWS integration
cm-cloud-configure --provider aws --region us-west-2
cm-cloud-credentials --access-key AKIAXXXXX --secret-key xxxxx
cm-cloud-template --instance-type p3.8xlarge --image ami-xxxxx

# Auto-scaling rules
cm-autoscale-enable --min-nodes 0 --max-nodes 10
cm-autoscale-rule --queue-length 5 --scale-up 2
cm-autoscale-rule --idle-time 600 --scale-down 1

# Burst job submission
sbatch --partition=cloud --constraint=aws gpu-training.sh
```

Multi-Cloud Management

```bash
# Multiple cloud providers
cm-cloud-add --name azure --provider azure --region eastus
cm-cloud-add --name gcp --provider gcp --region us-central1

# Workload distribution
cm-scheduler-policy --prefer on-premise --fallback cloud
cm-cost-optimize --provider cheapest --constraint gpu-memory=32GB
```

---

8. Backup & Disaster Recovery

Configuration Backup

```bash
# Automated backup configuration
cm-backup-configure --schedule daily --retention 30
cm-backup-include --path /etc/base-command-manager
cm-backup-include --path /home --exclude "*.tmp,*.cache"

# Manual backup
cm-backup-create --name "pre-upgrade-$(date +%Y%m%d)"
cm-backup-verify --name pre-upgrade-20231201

# Restore procedures
cm-backup-restore --name pre-upgrade-20231201 --target /tmp/restore
cm-cluster-restore --from /tmp/restore --components config,users,jobs
```

Disaster Recovery Procedures

```bash
```

```
# Head node failure recovery
# 1. Install BCM on replacement hardware
# 2. Restore from backup
cm-cluster-restore --full --backup latest

# 3. Update DNS and network configuration
cm-network-update --head-node-ip 192.168.1.100

# 4. Restart cluster services
cm-service-restart --all
cm-node-wake --all-compute-nodes
```

---

9. Capacity Planning & Scaling

Resource Utilization Analysis

```
bash

# Historical usage analysis
cm-usage-report --period monthly --format csv
cm-gpu-utilization --nodes all --timeframe "last 30 days"
cm-job-efficiency --users all --metrics gpu-hours,cpu-hours

# Capacity forecasting
cm-forecast --growth-rate 20% --horizon 12-months
cm-bottleneck-analysis --identify compute,storage,network
```

Scaling Procedures

```
bash
```

```
# Adding new compute nodes
cm-node-add-batch --hostnames compute[09-16] --image ai-workload-v2
cm-partition-expand --partition gpu --nodes compute[09-12]
cm-partition-expand --partition cpu --nodes compute[13-16]

# Storage scaling
cm-storage-expand --filesystem /scratch --capacity +100TB
cm-storage-rebalance --filesystem /scratch

# Network scaling
cm-network-upgrade --interface ib0 --speed 200Gb
cm-network-add --interface ib1 --topology fat-tree
```

---

10. Exam-Specific Scenarios

Scenario 1: GPU Allocation Issue

Problem: Jobs requesting GPUs remain in pending state

```
bash

# Diagnosis
squeue | grep PD  # Check pending jobs
sinfo -o "%P %a %l %D %T %N"  # Partition status
scontrol show partition gpu

# Solution steps
1. Check GPU availability: sinfo -o "%n %G"
2. Verify GRES configuration: scontrol show node compute-01
3. Update Slurm configuration if needed
4. Restart slurm services: systemctl restart slurmd
```

Scenario 2: Node Performance Degradation

Problem: Compute node showing poor performance

```
bash

# Systematic diagnosis
1. Check system resources: top, free -h, df -h
2. Monitor GPU health: nvidia-smi, dcgmi health -g 0
3. Check network: ibstat, ethtool eth0
4. Review system logs: journalctl -u kubelet -f

# Performance optimization
1. Update drivers: cm-driver-update --node compute-01
2. Optimize power settings: nvidia-smi -pl 300
3. Check thermal throttling: nvidia-smi -q -d TEMPERATURE
```

Scenario 3: Kubernetes Pod Scheduling Issues

Problem: Pods stuck in pending state despite available resources

```
bash

# Troubleshooting workflow
kubectl describe pod stuck-pod  # Check events
kubectl get nodes -o wide  # Node status
kubectl describe node compute-01  # Resource allocation

# Common fixes
1. Taint removal: kubectl taint nodes compute-01 key=value:NoSchedule-
2. Label addition: kubectl label nodes compute-01 gpu=true
3. Resource quota adjustment: kubectl edit resourcequota gpu-quota
```

This detailed guide provides the practical, hands-on procedures and troubleshooting steps essential for the NCP-AIOPS certification exam. Each section includes real commands, configuration examples, and step-by-step problem resolution procedures that system administrators encounter in production environments.