CloudForge Networking Guide

Document ID: PRD-CF-040 **Last Updated:** 2024-03-01 **Owner:** CloudForge Product Team **Classification:** Public

---

Overview

CloudForge provides comprehensive networking capabilities to build secure, scalable cloud infrastructure. This guide covers VPC configuration, load balancing, DNS, and network security.

Virtual Private Cloud (VPC)

Creating a VPC

```
# environment.yaml
networking:
  vpc:
    name: production-vpc
    cidr: 10.0.0.0/16
    enable_dns: true
    enable_dns_hostnames: true
    tags:
      environment: production
```

Subnets

```
networking:
  vpc:
    name: production-vpc
    cidr: 10.0.0.0/16
    subnets:
      public:
        - cidr: 10.0.1.0/24
          availability_zone: a
          map_public_ip: true
        - cidr: 10.0.2.0/24
          availability_zone: b
```

```
      map_public_ip: true
    private:
      - cidr: 10.0.10.0/24
        availability_zone: a
      - cidr: 10.0.11.0/24
        availability_zone: b
    database:
      - cidr: 10.0.20.0/24
        availability_zone: a
      - cidr: 10.0.21.0/24
        availability_zone: b
```

Internet Gateway

```
networking:
  vpc:
    internet_gateway:
      enabled: true
      routes:
        - destination: 0.0.0.0/0
          target: internet_gateway
          subnets: public
```

NAT Gateway

```
networking:
  vpc:
    nat_gateway:
      enabled: true
      type: single  # single, per_az
      routes:
        - destination: 0.0.0.0/0
          target: nat_gateway
          subnets: private
```

Security Groups

Defining Security Groups

```
networking:
  security_groups:
```

```yaml
- name: web-servers
  description: Security group for web servers
  ingress:
    - protocol: tcp
      port: 443
      source: 0.0.0.0/0
      description: HTTPS from internet
    - protocol: tcp
      port: 80
      source: 0.0.0.0/0
      description: HTTP from internet
  egress:
    - protocol: all
      destination: 0.0.0.0/0
      description: Allow all outbound

- name: api-servers
  description: Security group for API servers
  ingress:
    - protocol: tcp
      port: 8080
      source_security_group: web-servers
      description: From web servers
  egress:
    - protocol: tcp
      port: 5432
      destination_security_group: database
      description: To database

- name: database
  description: Security group for databases
  ingress:
    - protocol: tcp
      port: 5432
      source_security_group: api-servers
      description: PostgreSQL from API
```

Security Group Best Practices

1. **Least privilege** - Only allow necessary traffic
2. **Use security group references** - Instead of IP ranges where possible
3. **Separate by tier** - Different groups for web, app, database
4. **Document rules** - Use descriptions

Load Balancing

Application Load Balancer

```yaml
networking:
  load_balancers:
    - name: web-alb
      type: application
      scheme: internet-facing
      subnets: public
      security_groups:
        - web-servers
      listeners:
        - port: 443
          protocol: HTTPS
          certificate: arn:aws:acm:...:certificate/xxx
          default_action:
            type: forward
            target_group: web-targets
        - port: 80
          protocol: HTTP
          default_action:
            type: redirect
            redirect:
              protocol: HTTPS
              port: 443
              status_code: HTTP_301
```

Target Groups

```yaml
networking:
  target_groups:
    - name: web-targets
      port: 8080
      protocol: HTTP
      vpc: production-vpc
      health_check:
        path: /health
        interval: 30
        timeout: 5
        healthy_threshold: 2
        unhealthy_threshold: 3
      targets:
        - type: instance
          auto_scaling_group: web-asg
```

Network Load Balancer

```yaml
networking:
  load_balancers:
    - name: api-nlb
      type: network
      scheme: internal
      subnets: private
      listeners:
        - port: 443
          protocol: TLS
          certificate: arn:aws:acm:...:certificate/xxx
          target_group: api-targets
```

Path-Based Routing

```yaml
networking:
  load_balancers:
    - name: api-alb
      listeners:
        - port: 443
          protocol: HTTPS
          rules:
            - priority: 1
              conditions:
                - path_pattern: /api/v1/*
              action:
                type: forward
                target_group: api-v1-targets
            - priority: 2
              conditions:
                - path_pattern: /api/v2/*
              action:
                type: forward
                target_group: api-v2-targets
            - priority: 100
              conditions:
                - path_pattern: /*
              action:
                type: forward
                target_group: default-targets
```

DNS Management

Route 53 Integration

```yaml
networking:
  dns:
    hosted_zone: example.com
    records:
      - name: api.example.com
        type: A
        alias:
          target: web-alb
          evaluate_health: true
      - name: www.example.com
        type: A
        alias:
          target: cdn-distribution
      - name: db.internal.example.com
        type: CNAME
        value: database-endpoint.region.rds.amazonaws.com
        ttl: 300
```

Health Checks

```yaml
networking:
  dns:
    health_checks:
      - name: api-health
        type: HTTPS
        fqdn: api.example.com
        port: 443
        path: /health
        request_interval: 30
        failure_threshold: 3
```

Failover Routing

```yaml
networking:
  dns:
    records:
      - name: api.example.com
        type: A
        routing_policy: failover
        failover:
```

```
          primary:
            target: us-west-alb
            health_check: api-health-west
          secondary:
            target: us-east-alb
            health_check: api-health-east
```

Geolocation Routing

```
networking:
  dns:
    records:
      - name: app.example.com
        type: A
        routing_policy: geolocation
        geolocation:
          - location: US
            target: us-alb
          - location: EU
            target: eu-alb
          - location: default
            target: us-alb
```

CDN (Content Delivery Network)

CloudFront Distribution

```
networking:
  cdn:
    - name: static-cdn
      enabled: true
      origins:
        - domain: static-bucket.s3.amazonaws.com
          origin_id: s3-static
          s3_origin:
            origin_access_identity: true
      default_cache_behavior:
        target_origin: s3-static
        viewer_protocol_policy: redirect-to-https
        allowed_methods: [GET, HEAD]
        cached_methods: [GET, HEAD]
        ttl:
```

```
        default: 86400
        max: 31536000
      compress: true
    custom_error_responses:
      - error_code: 404
        response_code: 200
        response_page: /index.html
    aliases:
      - static.example.com
    certificate: arn:aws:acm:us-east-1:...:certificate/xxx
```

VPC Peering

Peering Configuration

```
networking:
  vpc_peering:
    - name: prod-to-shared
      requester_vpc: production-vpc
      accepter_vpc: shared-services-vpc
      auto_accept: true
      routes:
        - requester_cidr: 10.0.0.0/16
          accepter_cidr: 10.1.0.0/16
```

Cross-Account Peering

```
networking:
  vpc_peering:
    - name: cross-account-peer
      requester_vpc: production-vpc
      accepter:
        account_id: "123456789012"
        vpc_id: vpc-abc123
        region: us-east-1
```

VPN Connections

Site-to-Site VPN

```yaml
networking:
  vpn:
    - name: office-vpn
      type: site-to-site
      customer_gateway:
        ip_address: 203.0.113.1
        bgp_asn: 65000
      virtual_private_gateway:
        vpc: production-vpc
      tunnel_options:
        - pre_shared_key: ${VPN_PSK_1}
        - pre_shared_key: ${VPN_PSK_2}
      routes:
        - destination: 192.168.0.0/16
```

Client VPN

```yaml
networking:
  vpn:
    - name: remote-access
      type: client
      client_cidr: 10.100.0.0/16
      vpc: production-vpc
      subnets: private
      authentication:
        type: certificate
        root_certificate: arn:aws:acm:...:certificate/xxx
      split_tunnel: true
      dns_servers:
        - 10.0.0.2
```

Network ACLs

NACL Configuration

```yaml
networking:
  network_acls:
    - name: public-nacl
```

```
    vpc: production-vpc
    subnets: public
    ingress:
      - rule_number: 100
        protocol: tcp
        port_range: 443
        cidr: 0.0.0.0/0
        action: allow
      - rule_number: 110
        protocol: tcp
        port_range: 80
        cidr: 0.0.0.0/0
        action: allow
      - rule_number: 200
        protocol: tcp
        port_range: 1024-65535
        cidr: 0.0.0.0/0
        action: allow
    egress:
      - rule_number: 100
        protocol: all
        cidr: 0.0.0.0/0
        action: allow
```

VPC Endpoints

Gateway Endpoints

```
networking:
  vpc_endpoints:
    - name: s3-endpoint
      type: gateway
      service: s3
      vpc: production-vpc
      route_tables: private

    - name: dynamodb-endpoint
      type: gateway
      service: dynamodb
      vpc: production-vpc
      route_tables: private
```

Interface Endpoints

```yaml
networking:
  vpc_endpoints:
    - name: secrets-manager-endpoint
      type: interface
      service: secretsmanager
      vpc: production-vpc
      subnets: private
      security_groups:
        - internal-endpoints
      private_dns: true
```

Traffic Mirroring

```yaml
networking:
  traffic_mirror:
    - name: security-analysis
      source:
        network_interface: eni-xxx
      target:
        type: network_load_balancer
        arn: arn:aws:elasticloadbalancing:...:loadbalancer/net/xxx
      filter:
        ingress:
          - protocol: tcp
            port_range: 443
            action: accept
```

Monitoring

VPC Flow Logs

```yaml
networking:
  flow_logs:
    - name: vpc-flow-logs
      vpc: production-vpc
      traffic_type: ALL
      destination:
```

```
type: cloudwatch
log_group: /vpc/flow-logs
retention_days: 30
```

Network Metrics

Available in DataLens: - Bytes in/out - Packets in/out - Connection counts - Latency metrics - Error rates

---

Best Practices

Security

1. **Use private subnets** for application and database tiers
2. **Implement least privilege** security groups
3. **Enable VPC Flow Logs** for auditing
4. **Use VPC endpoints** to avoid public internet
5. **Encrypt data in transit** with TLS

Performance

1. **Use multiple availability zones** for high availability
2. **Right-size NAT gateways** for throughput needs
3. **Enable connection draining** on load balancers
4. **Use CDN** for static content
5. **Monitor and optimize** network costs

Architecture

1. **Plan CIDR ranges** for future growth
2. **Use consistent naming** conventions
3. **Document network topology**
4. **Automate with infrastructure as code**
5. **Test failover scenarios**

Troubleshooting

Connectivity Issues

```
# Test connectivity
cloudforge network test-connectivity \
  --source instance-id \
  --destination ip-address \
  --port 443

# Analyze path
cloudforge network analyze-path \
  --source eni-xxx \
  --destination eni-yyy
```

Common Issues

Issue	Possible Causes	Solution
Cannot reach internet	NAT gateway missing, route table	Check routes, NAT config
Cannot reach private resource	Security group, NACL	Review SG and NACL rules
High latency	Cross-AZ traffic, undersized	Use same AZ, scale resources
Connection timeouts	Health check failures	Check target health, logs

---

Related Documents: Getting Started (PRD-CF-001), Security Guide (PRD-CF-045), Multi-Region (PRD-CF-015)