Foundations

Foundational apps for almost any Infinity Tools installation.

3: Infrastructure Prerequisites

This chapter covers the essential infrastructure components that must be in place before deploying applications with Infinity Tools. The system includes automated readiness checks, but understanding the underlying architecture is crucial for troubleshooting and optimization.

Prerequisites Overview

Infinity Tools requires the following infrastructure components, which it handles automatically:

- **Docker Engine** - Container runtime and orchestration
- **Docker Compose** - Multi-container application definitions
- **Docker Network** - Service discovery and communication
- **System Dependencies** - GUM, Dialog, curl/wget
- **Resource Requirements** - Disk space, memory, network

Automated Readiness Check

Infinity Tools includes an automated readiness checker that validates and installs prerequisites:

```
sudo infinity-tools
```

The readiness check performs the following operations in sequence:

1. System Requirements Validation

```
# Check root privileges
[ "$EUID" -eq 0 ]

# Verify disk space (minimum 1GB)
df / | awk 'NR==2 {print $4}'

# Check for download tools
command -v curl || command -v wget

# WSL2 detection and configuration
```

```
grep -qi microsoft /proc/version
```

2. GUM Installation

GUM provides the modern terminal UI. Installation process:

```
# Architecture detection
arch=$(uname -m)
case $arch in
    x86_64) arch="x86_64" ;;
    aarch64|arm64) arch="arm64" ;;
    armv7l) arch="armv7" ;;
esac

# Download and install binary
gum_version="v0.13.0"
download_url="https://github.com/charmbracelet/gum/releases/download/${gum_version}/gum_${gum_
version#v}_Linux_${arch}.tar.gz"
```

3. Docker Installation and Configuration

Docker installation is handled by the `install-docker.sh` script:

```
# Check existing installation
command -v docker

# Verify Docker service status
systemctl is-active docker

# Check Docker Compose availability
docker compose version || docker-compose --version
```

4. Docker Network Setup

Creates the default "proxy" network for service communication:

```
# Check existing networks
docker network ls --format '{{.Name}}' | grep -v -E '^(bridge|host|none)$'
```

```
# Create network if needed
docker network create proxy

# Store network configuration
echo "DOCKER_NETWORK=proxy" > /tmp/infinity-tools-network.conf
```

Docker Architecture

Container Runtime

Infinity Tools uses Docker Engine with the following configuration:

- **Storage Driver:** Overlay2 (default)
- **Network Driver:** Bridge networks
- **Logging Driver:** JSON-file
- **Restart Policy:** unless-stopped

Docker Compose Integration

All services are defined using Docker Compose v2 (plugin) format:

```
version: '3.8'

services:
  service-name:
    image: image:tag
    container_name: service-name
    restart: unless-stopped
    networks:
      - proxy
    volumes:
      - /opt/speedbits/service:/data
    environment:
      - KEY=value
```

Network Architecture

Default Network Configuration

The "proxy" network provides:

- **Service Discovery:** Containers can reach each other by name
- **Isolation:** Services are isolated from host network
- **Traefik Integration:** Enables automatic reverse proxy configuration
- **DNS Resolution:** Built-in container name resolution

Network Topology

```
Internet → Traefik (proxy network) → Application Containers
                                  ↓
                      Database Containers (borgmatic-db network)
```

System Dependencies

Required Packages

The readiness check installs the following packages:

```
# Core utilities
curl wget git jq

# UI components
gum dialog

# Optional tools
rclone
```

Package Manager Support

Infinity Tools supports multiple package managers:

- **APT** (Debian/Ubuntu)
- **DNF** (Fedora/RHEL)
- **YUM** (CentOS/RHEL)
- **PACMAN** (Arch Linux)

- **APK** (Alpine Linux)

Resource Requirements

Minimum Specifications

- **CPU:** 1 core (2+ recommended)
- **RAM:** 2GB (4GB+ recommended)
- **Storage:** 20GB SSD (50GB+ recommended)
- **Network:** Stable internet connection

Storage Considerations

Infinity Tools uses the following storage structure:

```
/opt/speedbits/
├── _configuration/          # Global configuration
├── traefik/                 # Traefik data and certificates
├── wordpress/               # WordPress instances
├── vaultwarden/             # Vaultwarden data
└── ...                      # Other service data

/var/lib/docker/             # Docker system data
├── volumes/                 # Named volumes
├── networks/                # Network configurations
└── containers/              # Container data
```

Security Considerations

Docker Security

- **Non-root containers:** Services run with PUID/PGID 1000
- **Read-only filesystems:** Where applicable
- **No new privileges:** Security opt applied
- **Network isolation:** Services on isolated networks

File Permissions

```
# Check script permissions
find /opt/InfinityTools -name "*.sh" -exec ls -la {} \;

# Verify ownership
ls -la /opt/InfinityTools/

# Check for world-writable files
find /opt/InfinityTools -perm -002 -type f
```

Monitoring and Troubleshooting

System Status Commands

```
# Docker status
systemctl status docker
docker info

# Network status
docker network ls
docker network inspect proxy

# Container status
docker ps -a
docker stats

# Logs
journalctl -u docker
docker logs container-name
```

Common Issues

Docker service not starting:

```
sudo systemctl start docker
sudo systemctl enable docker
sudo systemctl status docker
```

Network creation fails:

```
# Check if network already exists
docker network ls | grep proxy

# Remove and recreate if needed
docker network rm proxy
docker network create proxy
```

Permission issues:

```
# Add user to docker group
sudo usermod -aG docker $USER
newgrp docker

# Check group membership
groups $USER
```

Performance Optimization

Docker Configuration

For production environments, consider these optimizations:

```
# /etc/docker/daemon.json
{
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "10m",
    "max-file": "3"
  },
  "storage-driver": "overlay2",
  "storage-opts": [
    "overlay2.override_kernel_check=true"
  ]
}
```

Resource Limits

Set appropriate resource limits for containers:

```
services:
  service-name:
    deploy:
      resources:
        limits:
          memory: 512M
          cpus: '0.5'
        reservations:
          memory: 256M
          cpus: '0.25'
```

Backup Considerations

Docker Data Backup

Important data locations for backup:

- **Application Data:** `/opt/speedbits/`
- **Docker Volumes:** `/var/lib/docker/volumes/`
- **Configuration:** `/opt/InfinityTools/`
- **SSL Certificates:** `/opt/speedbits/traefik/letsencrypt/`

Next Steps

With infrastructure prerequisites satisfied, you're ready to deploy Traefik - the reverse proxy that provides SSL termination and routing for all services.

Verification Checklist

- □ Docker Engine running and accessible
- □ Docker Compose available
- □ Proxy network created
- □ GUM installed for UI
- □ System resources adequate
- □ Network connectivity verified

Next: Installing Traefik - Reverse Proxy Configuration (Chapter 4)

4. Traefik - Reverse Proxy Configuration

Traefik serves as the reverse proxy and SSL termination point for all Infinity Tools services. This chapter covers the installation, configuration, and management of Traefik within the Infinity Tools ecosystem. For advanced configuration and troubleshooting, refer to the official Traefik documentation.

Traefik Architecture Overview

Traefik provides the following core functionality:

- **SSL/TLS Termination** - Automatic Let's Encrypt certificate management
- **Reverse Proxy** - Request routing based on Host headers
- **Load Balancing** - Distribution of traffic across service instances
- **Service Discovery** - Automatic detection of Docker containers
- **Middleware Support** - Security headers, authentication, rate limiting

Installation Process

Via Infinity Tools Menu

Navigate to the Infinity Tools menu and select:

```
SECURITY & NETWORKING → Install Traefik
```

Command Line Installation

```
# Direct script execution
sudo bash /opt/InfinityTools/Solutions/setup-traefik.sh --install

# With environment variables
export ACME_EMAIL="admin@domain.com"
```

```
export PROXY_NETWORK="proxy"
sudo -E bash /opt/InfinityTools/Solutions/setup-traefik.sh --install
```

Configuration Parameters

Required Configuration

During installation, you'll be prompted for:

- **ACME Email:** Email address for Let's Encrypt certificate notifications
- **Domain Name:** Primary domain for SSL certificate generation
- **IPv6 Support:** Enable/disable IPv6 for ACME challenges
- **Network Configuration:** Docker network for service communication

Environment Variables

```
# Optional environment variables
export ACME_EMAIL="admin@domain.com"        # Let's Encrypt email
export PROXY_NETWORK="proxy"                # Docker network name
export TRAEFIK_DOMAIN="traefik.domain.com"  # Traefik dashboard domain
export TRAEFIK_PORT="8080"                  # Dashboard port (if enabled)
```

Generated Configuration

Traefik Configuration File

Location: `/opt/speedbits/traefik/traefik.yml`

```
entryPoints:
  web:
    address: ":80"
    http:
      redirections:
        entryPoint:
          to: websecure
          scheme: https
```

```
          permanent: true

  websecure:
    address: ":443"

certificatesResolvers:
  myresolver:
    acme:
      email: admin@domain.com
      storage: /letsencrypt/acme.json
      httpChallenge:
        entryPoint: web

providers:
  docker:
    exposedByDefault: false

serversTransport:
  insecureSkipVerify: true

global:
  checkNewVersion: false
  sendAnonymousUsage: false
```

Docker Compose Configuration

Location: `/opt/speedbits/traefik/docker-compose.yml`

```
version: '3.8'

services:
  traefik:
    image: traefik:v3.0
    container_name: traefik
    command:
      - "--configFile=/traefik.yml"
    ports:
      - "80:80"
      - "443:443"
```

```
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock:ro
      - /opt/speedbits/traefik/traefik.yml:/traefik.yml:ro
      - /opt/speedbits/traefik/letsencrypt:/letsencrypt
    restart: unless-stopped
    networks:
      - proxy

networks:
  proxy:
    external: true
```

SSL Certificate Management

Let's Encrypt Integration

Traefik automatically manages SSL certificates using Let's Encrypt:

- **HTTP-01 Challenge:** Validates domain ownership via HTTP
- **Automatic Renewal:** Certificates are renewed automatically
- **Wildcard Support:** Supports wildcard certificates via DNS challenge
- **Certificate Storage:** Stored in `/opt/speedbits/traefik/letsencrypt/`

Certificate Monitoring

```
# Check certificate status
docker logs traefik | grep -i acme

# View certificate files
ls -la /opt/speedbits/traefik/letsencrypt/

# Check certificate expiration
openssl x509 -in /opt/speedbits/traefik/letsencrypt/acme.json -text -noout
```

Service Integration

Automatic Service Discovery

Traefik automatically discovers services with the following labels:

```
labels:
  - "traefik.enable=true"
  - "traefik.http.routers.service-name.rule=Host(`service.domain.com`)"
  - "traefik.http.routers.service-name.entrypoints=websecure"
  - "traefik.http.routers.service-name.tls.certresolver=myresolver"
  - "traefik.http.services.service-name.loadbalancer.server.port=8080"
```

Network Requirements

Services must be connected to the same Docker network as Traefik:

```
networks:
  - proxy

# Ensure network exists
docker network create proxy
```

Security Configuration

Security Headers

Traefik can be configured with security middleware:

```
labels:
  - "traefik.http.middlewares.security-headers.headers.customResponseHeaders.X-Content-Type-Options=nosniff"
  - "traefik.http.middlewares.security-headers.headers.customResponseHeaders.X-Frame-Options=SAMEORIGIN"
  - "traefik.http.middlewares.security-headers.headers.customResponseHeaders.X-XSS-Protection=1; mode=block"
  - "traefik.http.routers.service-name.middlewares=security-headers"
```

Access Control

Basic authentication can be configured for services:

```
# Generate password hash
echo $(htpasswd -nb admin password) | sed -e s/\\$/\\$\\$/g

# Apply to service
labels:
  - "traefik.http.middlewares.auth.basicauth.users=admin:$$2y$$10$$..."
  - "traefik.http.routers.service-name.middlewares=auth"
```

Monitoring and Logging

Traefik Dashboard

The Traefik dashboard can be enabled for monitoring:

```
# Add to docker-compose.yml
labels:
  - "traefik.enable=true"
  - "traefik.http.routers.dashboard.rule=Host(`traefik.domain.com`)"
  - "traefik.http.routers.dashboard.tls.certresolver=myresolver"
  - "traefik.http.routers.dashboard.service=api@internal"
```

Logging Configuration

```
# Add to traefik.yml
log:
  level: INFO
  filePath: /var/log/traefik.log

accessLog:
  filePath: /var/log/access.log
  format: json
```

Performance Optimization

Resource Limits

```
services:
  traefik:
    deploy:
      resources:
        limits:
          memory: 512M
          cpus: '0.5'
        reservations:
          memory: 256M
          cpus: '0.25'
```

Caching Configuration

```
# Add to traefik.yml
http:
  middlewares:
    cache:
      headers:
        customRequestHeaders:
          Cache-Control: "max-age=3600"
```

Troubleshooting

Common Issues

Certificate Generation Fails:

```
# Check domain DNS resolution
dig domain.com
nslookup domain.com

# Verify port 80 accessibility
telnet domain.com 80

# Check Traefik logs
```

```
docker logs traefik | grep -i acme
```

Service Not Accessible:

```
# Check service labels
docker inspect service-name | grep -A 10 Labels

# Verify network connectivity
docker network inspect proxy

# Check Traefik routing
curl -H "Host: service.domain.com" http://localhost
```

Performance Issues:

```
# Monitor resource usage
docker stats traefik

# Check connection limits
ss -tulnp | grep :443

# Review access logs
tail -f /opt/speedbits/traefik/logs/access.log
```

Debugging Commands

```
# Check Traefik configuration
docker exec traefik traefik version

# Test configuration
docker exec traefik traefik --configFile=/traefik.yml --logLevel=DEBUG

# View active routes
curl -s http://localhost:8080/api/http/routers | jq

# Check certificate status
docker exec traefik cat /letsencrypt/acme.json | jq
```

Backup and Recovery

Configuration Backup

```
# Backup Traefik configuration
tar -czf traefik-backup.tar.gz -C /opt/speedbits/traefik .

# Backup SSL certificates
cp -r /opt/speedbits/traefik/letsencrypt/ /backup/traefik-certs/
```

Disaster Recovery

```
# Restore configuration
tar -xzf traefik-backup.tar.gz -C /opt/speedbits/traefik/

# Restart Traefik
cd /opt/speedbits/traefik
docker compose down
docker compose up -d
```

Integration with Other Services

Service Dependencies

Most Infinity Tools services check for Traefik availability:

```
# Service installation checks
if ! docker ps --format '{{.Names}}' | grep -q "^traefik$"; then
    echo "Traefik is not running!"
    echo "Please install Traefik first"
    exit 1
fi
```

Network Integration

Services automatically join the proxy network:

```
networks:
  proxy:
    external: true
    name: proxy
```

Advanced Configuration

Custom Middleware

```
# Rate limiting
labels:
  - "traefik.http.middlewares.ratelimit.ratelimit.burst=100"
  - "traefik.http.middlewares.ratelimit.ratelimit.average=50"

# IP whitelisting
labels:
  - "traefik.http.middlewares.ipwhitelist.ipwhitelist.sourcerange=192.168.1.0/24"
```

Load Balancing

```
labels:
  - "traefik.http.services.service-name.loadbalancer.server.port=8080"
  - "traefik.http.services.service-name.loadbalancer.healthcheck.path=/health"
  - "traefik.http.services.service-name.loadbalancer.healthcheck.interval=30s"
```

Next Steps

With Traefik installed and configured, you can now deploy applications that will automatically integrate with the reverse proxy system.

Verification Checklist

- [ ] Traefik container running and healthy
- [ ] SSL certificates generated for configured domains
- [ ] HTTP to HTTPS redirection working
- [ ] Docker network connectivity verified

- Service discovery functioning

Next: Application Deployment and Management (Coming Soon)

5: Apprise - Notifications Hub

Apprise provides a unified notification gateway (HTTP API) for 90+ providers (email, Slack, Discord, Telegram, etc.). Infinity Tools integrates Apprise for infrastructure alerts (e.g., Borgmatic). For provider matrices and syntax, see the official Apprise documentation.

We included it here since Borgmatic, the backup solution provided by Infinity Tools, requires it (and we assume you like to create backups).

Prerequisites

- □ Traefik installed (Chapter 4) for HTTPS (optional)
- □ Docker running (Chapter 3)
- □ Optional: Subdomain (Chapter 4.5), e.g., `alerts.example.com`

Installation via Infinity Tools

Menu Installation

```
□□APPLICATIONS → Apprise → Install
```

CLI Installation

```
sudo bash /opt/InfinityTools/Solutions/setup-apprise.sh --install
# With domain (Traefik mode)
export APPRISE_DOMAIN="alerts.example.com"
sudo -E bash /opt/InfinityTools/Solutions/setup-apprise.sh --install
```

Service Endpoints

- Local API: `http://apprise:8000/notify`
- With domain (Traefik): `https://alerts.example.com/notify`

Provider URLs

Apprise uses provider URLs to define targets:

```
# SMTP (STARTTLS)
mailtos://USERNAME:PASSWORD@SMTP_HOST:587/?from=from@example.com&to=ops@example.com

# Slack (Webhook)
slack://TOKENA/TOKENB/TOKENC

# Telegram
tgram://BOT_TOKEN/CHAT_ID
```

Sending Notifications

cURL

```
curl -X POST "http://apprise:8000/notify" \
  -d "title=Backup" \
  -d "body=Borgmatic completed successfully" \
  -d "url=PROVIDER_URL"
```

JSON

```
curl -X POST "http://apprise:8000/notify" \
  -H 'Content-Type: application/json' \
  -d '{
    "title": "Backup",
    "body": "Borgmatic completed successfully",
    "url": ["PROVIDER_URL1", "PROVIDER_URL2"]
  }'
```

Integration Notes

- **Borgmatic:** Points to Apprise API for start/finish/failure hooks

- **Other services:** Can post to the same endpoint
- **Security:** Prefer Traefik HTTPS and restrict access if exposing publicly

Troubleshooting

- Logs: `docker logs apprise`
- Validate provider URL syntax against docs
- Check egress connectivity to provider endpoints

Next

Proceed to Borgmatic (Chapter 6) to configure automated backups with notifications.

6: Borgmatic - Enterprise Backup Solution

Borgmatic is a configuration-driven wrapper for BorgBackup that provides enterprise-grade backup capabilities with deduplication, encryption, and automated scheduling. It implements a dual-backup strategy separating file and database backups with different retention policies. For comprehensive configuration options, advanced features, and API documentation, please refer to the official Borgmatic documentation.

Interdependencies

Required dependency: Apprise must be installed and running for notification delivery (success/failure/security alerts). Install via □□ *APPLICATIONS → Apprise → Install* before enabling Borgmatic.

Prerequisites

- □ **Docker running** (Chapter 3)
- □ **Apprise installed** (Chapter 5) for notifications
- □ **Storage capacity** (backups require free disk space)
- □ **Optional:** Traefik (not required for Borgmatic)

Architecture Overview

Borgmatic in Infinity Tools implements a sophisticated backup architecture:

- □ **Dual Backup Strategy** - Separate file and database backup schedules
- □ **Deduplication** - Efficient storage using Borg's chunking algorithm
- □ **Encryption** - AES-256 encryption with repokey mode
- □ **Ransomware Protection** - Canary file monitoring system
- □ **Multi-Database Support** - MariaDB, PostgreSQL, SQLite, MongoDB
- □ **Notification Integration** - Apprise integration for alerts

Installation Methods

Via Infinity Tools Menu

Navigate to the Infinity Tools menu and select:

```
□□BACKUP MANAGEMENT → Install Borgmatic
```

Command Line Installation

```
# Direct script execution
sudo bash /opt/InfinityTools/Solutions/setup-borgmatic.sh --install

# With environment variables
export BORGMATIC_SCHEDULE="daily"
export BORGMATIC_RETENTION="7,4,6"
export BORGMATIC_COMPRESSION="zstd"
export BORGMATIC_ENCRYPTION="repokey"
sudo -E bash /opt/InfinityTools/Solutions/setup-borgmatic.sh --install
```

Configuration Parameters

Schedule Configuration

Borgmatic supports multiple schedule options:

```
# Environment Variables
export BORGMATIC_SCHEDULE="daily"        # daily, twice-daily, weekly
export BORGMATIC_RETENTION="7,4,6"       # daily,weekly,monthly
export BORGMATIC_COMPRESSION="zstd"      # lz4, zlib, lzma, zstd
export BORGMATIC_ENCRYPTION="repokey"    # none, keyfile, repokey
```

Dual Backup Strategy

The system implements two separate backup configurations:

- **File Backups:** Daily schedule, long-term retention
- **Database Backups:** High-frequency schedule, short-term retention

Generated Configuration

File Backup Configuration

Location: `/opt/speedbits/borgmatic/borgmatic-files.yml`

```
# Borgmatic Files Configuration
source_directories:
    - /opt/speedbits

repositories:
    - path: /backups/borgmatic-repo
      label: speedbits-repo

exclude_patterns:
    - '*.tmp'
    - '*.log'
    - '*/logs/*'
    - '*/cache/*'
    - '*/tmp/*'
    - '*/.git/*'
    - '*/node_modules/*'
    - '*/venv/*'
    - '*/__pycache__/*'
    - '*/database-dumps/*'
    - '*/netdata/lib/*'
    - '*/netdata/cache/*'

compression: zstd
archive_name_format: 'speedbits-files-{hostname}-{now:%Y-%m-%d-%H%M%S}'

keep_daily: 7
keep_weekly: 4
keep_monthly: 6

checks:
    - name: repository
      frequency: 2 weeks
```

```
    - name: archives
      frequency: 2 weeks
```

Database Backup Configuration

Location: `/opt/speedbits/borgmatic/borgmatic-databases.yml`

```
# Borgmatic Database Configuration
source_directories:
    - /backups/database-dumps

repositories:
    - path: /backups/borgmatic-repo
      label: speedbits-repo

compression: zstd,9
archive_name_format: 'speedbits-databases-{hostname}-{now:%Y-%m-%d-%H%M%S}'

keep_hourly: 48
keep_daily: 7

# Database preparation hooks
hooks:
    before_backup:
        - /usr/local/bin/backup-databases.sh
    after_backup:
        - /usr/local/bin/cleanup-database-dumps.sh
```

Security Features

Ransomware Protection

Borgmatic implements canary file monitoring:

```
# Canary file system
mkdir -p /opt/speedbits/dont-touch-this-folder
echo "This is a test" > /opt/speedbits/dont-touch-this-folder/dont-change-this-file-critical-
```

```
data.txt
chmod 644 /opt/speedbits/dont-touch-this-folder/dont-change-this-file-critical-data.txt

# Security monitoring in borgmatic-files.yml
commands:
    - before: action
      when: [create]
      run:
          - if [ ! -f /opt/speedbits/dont-touch-this-folder/dont-change-this-file-critical-
data.txt ]; then echo "CANARY FILE MISSING!" && curl -X POST http://apprise:8000/notify -d
"body=SECURITY ALERT Canary file is MISSING." -d "title=CANARY FILE MISSING" -d "tag=security"
2>/dev/null || true && exit 1; fi
```

Encryption Configuration

Borgmatic uses repokey encryption mode:

```
# Repository initialization
borg init --encryption=repokey --make-parent-dirs /backups/borgmatic-repo

# Passphrase management
BORG_PASSCOMMAND="cat /etc/borgmatic/repo-passphrase.txt"
export BORG_PASSCOMMAND
```

Database Integration

Multi-Database Support

Borgmatic automatically discovers and backs up multiple database types:

```
# Database discovery script
#!/bin/bash
# /usr/local/bin/backup-databases.sh

# MariaDB/MySQL
if docker ps --format '{{.Names}}' | grep -q "mariadb\|mysql"; then
    docker exec mariadb mysqldump --all-databases > /backups/database-dumps/mariadb-$(date
```

```
+%Y%m%d-%H%M%S).sql
fi

# PostgreSQL
if docker ps --format '{{.Names}}' | grep -q "postgres"; then
    docker exec postgres pg_dumpall > /backups/database-dumps/postgres-$(date +%Y%m%d-
%H%M%S).sql
fi

# SQLite
find /opt/speedbits -name "*.db" -o -name "*.sqlite" -o -name "*.sqlite3" | while read db; do
    cp "$db" "/backups/database-dumps/sqlite-$(basename "$db")-$(date +%Y%m%d-%H%M%S)"
done
```

Docker Compose Configuration

Container Setup

Location: `/opt/speedbits/borgmatic/docker-compose.yml`

```
version: '3.8'

services:
  borgmatic:
    image: borgmatic/borgmatic:latest
    container_name: borgmatic
    restart: unless-stopped
    environment:
      BORG_PASSCOMMAND: "cat /etc/borgmatic/repo-passphrase.txt"
    volumes:
      - /opt/speedbits:/opt/speedbits:ro
      - /opt/speedbits-backup:/backups
      - /opt/speedbits/borgmatic:/etc/borgmatic:ro
      - /var/run/docker.sock:/var/run/docker.sock:ro
    networks:
      - borgmatic-network
      - borgmatic-db
```

```
    command: |
      /bin/sh -c "
        echo 'Initializing Borgmatic...'

        # Install rclone for cloud storage support
        if ! command -v rclone >/dev/null 2>&1; then
          echo 'Installing rclone for cloud storage support...'
          apk add --no-cache rclone || echo 'rclone installation failed, continuing...'
        fi

        # Initialize repository if needed
        if ! borg info /backups/borgmatic-repo >/dev/null 2>&1; then
          echo 'Creating new Borg repository...'
          borg init --encryption=repokey --make-parent-dirs /backups/borgmatic-repo
        fi

        # Set up cron jobs
        {
          echo 'BORG_PASSCOMMAND=\"cat /etc/borgmatic/repo-passphrase.txt\"'
          echo '0 2 * * * borgmatic --config /etc/borgmatic/borgmatic-files.yml create --
verbosity 1'
          echo '0 */6 * * * borgmatic --config /etc/borgmatic/borgmatic-databases.yml create
--verbosity 1'
        } | crontab -

        # Start cron daemon
        crond -f
      "

networks:
  borgmatic-network:
    driver: bridge
  borgmatic-db:
    external: true
```

Monitoring and Notifications

Apprise Integration

Borgmatic integrates with Apprise for notifications:

```
# Notification configuration in borgmatic-files.yml
commands:
    - before: action
      when: [create]
      run:
          - curl -X POST http://apprise:8000/notify -d "body=Starting scheduled file backup" -
d "title=File Backup Started" -d "tag=backup" 2>/dev/null || true

    - after: action
      when: [create]
      states: [finish]
      run:
          - curl -X POST http://apprise:8000/notify -d "body=File backup completed
successfully" -d "title=File Backup Complete" -d "tag=backup" 2>/dev/null || true

    - after: action
      when: [create]
      states: [fail]
      run:
          - curl -X POST http://apprise:8000/notify -d "body=File backup FAILED" -d
"title=File Backup FAILED" -d "tag=backup,error" 2>/dev/null || true
```

Health Monitoring

```
# Check backup status
docker exec borgmatic borg list /backups/borgmatic-repo

# Check repository integrity
docker exec borgmatic borg check /backups/borgmatic-repo

# View backup logs
docker logs borgmatic

# Check cron jobs
docker exec borgmatic crontab -l
```

Advanced Configuration

Cloud Storage Integration

Borgmatic supports cloud storage via rclone:

```
# rclone configuration
rclone config create remote s3 \
  provider=AWS \
  access_key_id=your_access_key \
  secret_access_key=your_secret_key \
  region=us-east-1

# Cloud backup command
rclone sync /opt/speedbits-backup/borgmatic-repo remote:backups/borgmatic-repo
```

Custom Hooks

Implement custom backup hooks:

```
# Custom pre-backup hook
hooks:
  before_backup:
    - /usr/local/bin/pre-backup.sh
    - /usr/local/bin/backup-databases.sh
  after_backup:
    - /usr/local/bin/post-backup.sh
    - /usr/local/bin/cleanup-temp-files.sh
  on_error:
    - /usr/local/bin/backup-error-handler.sh
```

Performance Optimization

Compression Settings

Choose compression based on your needs:

- **lz4:** Fastest, least compression
- **zstd:** Balanced speed and compression
- **zlib:** Good compression, moderate speed
- **lzma:** Best compression, slowest

Resource Management

```
# Resource limits in docker-compose.yml
services:
  borgmatic:
    deploy:
      resources:
        limits:
          memory: 1G
          cpus: '1.0'
        reservations:
          memory: 512M
          cpus: '0.5'
```

Disaster Recovery

Backup Restoration

```
# List available archives
docker exec borgmatic borg list /backups/borgmatic-repo

# Extract specific archive
docker exec borgmatic borg extract /backups/borgmatic-repo::speedbits-files-server-2024-01-15-020000

# Extract to specific location
docker exec borgmatic borg extract /backups/borgmatic-repo::speedbits-files-server-2024-01-15-020000 /restore/path
```

Repository Recovery

```
# Check repository integrity
docker exec borgmatic borg check /backups/borgmatic-repo

# Repair repository if needed
docker exec borgmatic borg check --repair /backups/borgmatic-repo

# Recover from cloud storage
rclone sync remote:backups/borgmatic-repo /opt/speedbits-backup/borgmatic-repo
```

Troubleshooting

Common Issues

Backup Failures:

```
# Check container logs
docker logs borgmatic

# Check cron jobs
docker exec borgmatic crontab -l

# Test manual backup
docker exec borgmatic borgmatic --config /etc/borgmatic/borgmatic-files.yml create --verbosity
2
```

Repository Issues:

```
# Check repository status
docker exec borgmatic borg info /backups/borgmatic-repo

# Verify passphrase
docker exec borgmatic cat /etc/borgmatic/repo-passphrase.txt

# Check disk space
df -h /opt/speedbits-backup
```

Debug Commands

```
# Container status
docker ps | grep borgmatic

# Container exec
docker exec -it borgmatic /bin/sh

# Check mounted volumes
docker inspect borgmatic | grep -A 10 Mounts

# View configuration
docker exec borgmatic cat /etc/borgmatic/borgmatic-files.yml
```

Best Practices

Security

- Use strong, unique passphrases
- Store passphrases securely (password manager)
- Enable canary file monitoring
- Regular backup testing and restoration

Operational

- Monitor backup success/failure notifications
- Regular repository integrity checks
- Test disaster recovery procedures
- Document restoration procedures

Next Steps

With Borgmatic installed and configured, you have enterprise-grade backup protection ready for all your services. This infrastructure will automatically protect any applications you install going forward.

Verification Checklist

- □ Borgmatic container running and healthy
- □ File and database backup schedules active

- Repository initialized and encrypted
- Canary file monitoring enabled
- Notification system configured
- Backup restoration tested

Next: Installing Vaultwarden - Password Management Solution (Chapter 6)

7: Portainer - Docker Management Platform

Portainer CE (Community Edition) provides a web-based management interface for Docker environments. It offers container lifecycle management, stack deployment, image/volume/network management, resource monitoring, and role-based access control. For advanced features, API documentation, and enterprise features, see the official Portainer documentation.

Prerequisites

- **Docker installed** (Chapter 3)
- **Docker Compose** (Chapter 3)
- **Optional: Traefik installed** (Chapter 4) for HTTPS with Let's Encrypt
- **Optional: Domain configured** (Chapter 4.5), e.g., `portainer.example.com`

Installation via Infinity Tools

Menu Installation

```
APPLICATIONS → Portainer → Install
```

CLI Installation

```
sudo bash /opt/InfinityTools/Solutions/setup-portainer.sh --install


# With domain (Traefik mode)
export PORTAINER_DOMAIN="portainer.example.com"
sudo -E bash /opt/InfinityTools/Solutions/setup-portainer.sh --install
```

Deployment Modes

Traefik Mode (Recommended)

Uses Traefik for SSL termination and domain routing:

- Automatic Let's Encrypt certificate provisioning
- Domain-based access: `https://portainer.example.com`
- Security headers configured via Traefik middleware
- Requires: Traefik running, DNS A record configured

Standalone Mode

Direct HTTPS access with self-signed certificate:

- Access via: `https://SERVER_IP:9443`
- Self-signed SSL (browser warning on first access)
- No domain required
- Suitable for internal/development use

Installation Process

Configuration Steps

1. **SSL Mode Selection:** Choose Traefik or Standalone
2. **If Traefik:** Provide domain name (e.g., `portainer.example.com`)
3. **If Standalone:** Specify HTTPS port (default: 9443)
4. **Network Detection:** Automatically detects Traefik network if available

What Gets Created

- **Directory:** `/opt/speedbits/portainer`
- **Container:** `portainer` (portainer/portainer-ce:2.21.4)
- **Data Volume:** `/opt/speedbits/portainer/data`
- **Docker Compose:** `/opt/speedbits/portainer/docker-compose.yml`
- **Network:** Joins Traefik network (Traefik mode) or creates internal network (Standalone)

First-Time Setup

Admin Account Creation

⚠ **CRITICAL:** Portainer requires admin credential creation on first access. No default credentials exist.

1. Access Portainer via the provided URL
2. Create administrator account:
   - Username: Any (typically "admin")
   - Password: Minimum 12 characters (enforced)
3. Select Docker environment
4. Connect to local Docker socket

Password Recovery

If admin password is lost, reset requires:

```
cd /opt/speedbits/portainer
docker compose down
rm -rf data/
docker compose up -d
```

Note: This resets all Portainer configuration (users, settings, RBAC). Docker containers are unaffected.

Service Endpoints

- **Traefik mode:** `https://portainer.example.com`
- **Standalone mode:** `https://SERVER_IP:9443`
- **Internal API:** Portainer exposes port 9000 internally (mapped via Traefik or directly)

Key Features

Container Management

- Lifecycle operations (start/stop/restart/remove)
- Log viewing and streaming
- Container inspection (env vars, mounts, networks)
- Console access (exec into containers)
- Resource limits and constraints

Stack Deployment

- Deploy docker-compose stacks via web UI
- Edit stack configurations
- Update and rollback stacks
- Environment variable management

Image Management

- Browse local images
- Pull from registries (Docker Hub, private registries)
- Remove unused images
- Image tagging and management

Volume and Network Management

- Create/manage Docker volumes
- Network configuration and inspection
- Volume backup considerations

Monitoring and Statistics

- Real-time resource usage (CPU, memory, network)
- Container health status
- Historical performance data
- Event logs

Security Configuration

Initial Security

- □ Strong admin password (20+ characters recommended)
- □ Enable 2FA: Settings → Users → Two-Factor Authentication
- □ Use Traefik mode for production (Let's Encrypt SSL)
- □ Security headers configured via Traefik middleware (Traefik mode)

Role-Based Access Control (RBAC)

- Create team members with restricted access
- Assign roles (admin, operator, viewer)
- Environment-level permissions
- Audit logging

Backup Configuration

- Export Portainer settings: Settings → Backup Configuration
- Regular backups recommended
- Backup includes: users, roles, settings, stack definitions
- Does NOT include: Docker container data (managed separately via Borgmatic)

Integration with Infinity Tools

Portainer complements Infinity Tools by providing:

- Visual management of Infinity Tools containers
- Log access for troubleshooting
- Resource monitoring for capacity planning
- Stack deployment for custom applications

Note: Infinity Tools applications are managed via their respective setup scripts. Portainer provides visibility and operational control, but configuration changes should be made through Infinity Tools scripts to maintain consistency.

Troubleshooting

Container Not Starting

```
docker logs portainer
docker ps -a | grep portainer
```

Traefik Routing Issues

- Verify Traefik is running: `docker ps | grep traefik`
- Check Traefik logs: `docker logs traefik`
- Verify DNS resolution: `dig portainer.example.com`
- Confirm SSL certificate: Check Traefik dashboard or logs

Docker Socket Access

Portainer requires read-only access to `/var/run/docker.sock`. The setup script configures this automatically. If issues occur:

```
ls -l /var/run/docker.sock
docker ps  # Verify Docker is accessible
```

Production Considerations

- **High Availability:** Portainer CE is single-instance. For HA, consider Portainer Business Edition
- **Backup Strategy:** Regular Portainer config exports + Borgmatic for container data
- **Access Control:** Implement RBAC for team members
- **Network Security:** Use Traefik with firewall rules restricting access
- **Monitoring:** Integrate with monitoring solutions (Netdata, Uptime Kuma)

Next Steps

Portainer is now operational. Use it to:

- Monitor Infinity Tools containers
- Deploy additional Docker applications
- Manage Docker resources
- Troubleshoot container issues

For advanced Portainer features, team management, and enterprise capabilities, refer to the official Portainer documentation.