1.0.0 - ami-0ad4b852fd26dc0df

Mailcow Email Server - AWS Marketplace AMI

Complete, production-ready email server solution powered by Mailcow, optimized for AWS deployment.

Overview

This AMI provides a fully-featured email server including:

- **Postfix** - Mail Transfer Agent (MTA)
- **Dovecot** - IMAP/POP3 server
- **SOGo** - Webmail, calendar, and contacts
- **Rspamd** - Spam filtering
- **ClamAV** - Antivirus scanning
- **Let's Encrypt** - Automatic SSL certificates
- **Fail2ban** - Intrusion prevention (host-level for SSH)

Requirements

Instance Requirements

Component	Minimum	Recommended
Instance Type	t3.medium	t3.large or better
vCPUs	2	4+
RAM	4 GB	8 GB+
Root Volume	20 GB	30 GB
Data Volume (EBS)	50 GB	100 GB+

Network Requirements

Port	Protocol	Purpose
22	TCP	SSH
25	TCP	SMTP (inbound)
80	TCP	HTTP (Let's Encrypt)
110	TCP	POP3
143	TCP	IMAP
443	TCP	HTTPS
465	TCP	SMTPS
587	TCP	Submission
993	TCP	IMAPS
995	TCP	POP3S
4190	TCP	ManageSieve

DNS Requirements

Before configuration, set up these DNS records:

```
# A Record (required)
mail.example.com.  IN  A  YOUR_PUBLIC_IP

# MX Record (required)
example.com.  IN  MX  10  mail.example.com.

# SPF Record (required)
example.com.  IN  TXT  "v=spf1 mx a -all"

# DMARC Record (recommended)
_dmarc.example.com.  IN  TXT  "v=DMARC1; p=quarantine; rua=mailto:postmaster@example.com"

# DKIM Record (configure after installation via admin panel)
```

?? IMPORTANT: AWS Port 25 Restriction

AWS blocks outbound SMTP (port 25) by default. Your server cannot send emails until you:

1. **Request port 25 removal from AWS:**
   https://aws.amazon.com/forms/ec2-email-limit-rdns-request
2. **Or configure Amazon SES as relay**

See `/opt/mailcow/scripts/AWS-PORT25-GUIDE.md` for detailed instructions.

Quick Start

Option 1: Interactive Configuration (SSH)

1. **Launch the AMI** with:
   - Instance type: t3.medium or larger
   - Attach an EBS volume (50GB+) for mail data
   - Configure Security Group with required ports
2. **SSH into the instance:**

   ```
   ssh -i your-key.pem ubuntu@your-instance-ip
   ```

3. **Run the configuration script:**

   ```
   sudo /opt/mailcow/scripts/configure-mailcow.sh
   ```

4. **Follow the prompts** to configure:
   - Mail hostname (e.g., mail.example.com)
   - Admin email address
   - Timezone
   - SSL/TLS settings
5. **Access the admin panel:**
   - URL: `https://your-hostname`
   - Username: `admin`
   - Password: `moohoo` (change immediately!)

Option 2: Automated Configuration (User-Data)

Launch the instance with this JSON user-data:

```
{
  "hostname": "mail.example.com",
  "admin_email": "admin@example.com",
  "timezone": "UTC",
  "skip_letsencrypt": "n"
}
```

CloudFormation Example:

```
AWSTemplateFormatVersion: '2010-09-09'
Description: Mailcow Email Server

Parameters:
  MailHostname:
    Type: String
    Description: Mail server hostname (e.g., mail.example.com)
  AdminEmail:
    Type: String
    Description: Admin email for notifications and certificates
  InstanceType:
    Type: String
    Default: t3.medium
  KeyName:
    Type: AWS::EC2::KeyPair::KeyName

Resources:
  MailcowInstance:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref MailcowAMI  # Replace with actual AMI ID
      InstanceType: !Ref InstanceType
      KeyName: !Ref KeyName
      SecurityGroupIds:
        - !Ref MailcowSecurityGroup
      UserData:
        Fn::Base64: !Sub |
          {
            "hostname": "${MailHostname}",
            "admin_email": "${AdminEmail}",
            "timezone": "UTC"
```

```
        }
    BlockDeviceMappings:
      - DeviceName: /dev/sdf
        Ebs:
          VolumeSize: 100
          VolumeType: gp3
          DeleteOnTermination: false

MailcowSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Mailcow Email Server
    SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 22
        ToPort: 22
        CidrIp: 0.0.0.0/0
      - IpProtocol: tcp
        FromPort: 25
        ToPort: 25
        CidrIp: 0.0.0.0/0
      - IpProtocol: tcp
        FromPort: 80
        ToPort: 80
        CidrIp: 0.0.0.0/0
      - IpProtocol: tcp
        FromPort: 110
        ToPort: 110
        CidrIp: 0.0.0.0/0
      - IpProtocol: tcp
        FromPort: 143
        ToPort: 143
        CidrIp: 0.0.0.0/0
      - IpProtocol: tcp
        FromPort: 443
        ToPort: 443
        CidrIp: 0.0.0.0/0
      - IpProtocol: tcp
        FromPort: 465
        ToPort: 465
```

```
          CidrIp: 0.0.0.0/0
        - IpProtocol: tcp
          FromPort: 587
          ToPort: 587
          CidrIp: 0.0.0.0/0
        - IpProtocol: tcp
          FromPort: 993
          ToPort: 993
          CidrIp: 0.0.0.0/0
        - IpProtocol: tcp
          FromPort: 995
          ToPort: 995
          CidrIp: 0.0.0.0/0
        - IpProtocol: tcp
          FromPort: 4190
          ToPort: 4190
          CidrIp: 0.0.0.0/0
```

Management Commands

Use the `mailcow-cli` utility for common operations:

```
# Check service status
mailcow-cli status

# View logs
mailcow-cli logs
mailcow-cli logs postfix-mailcow  # Specific service

# Start/Stop/Restart
sudo mailcow-cli start
sudo mailcow-cli stop
sudo mailcow-cli restart

# Show configuration
mailcow-cli info

# Show required DNS records
```

```
mailcow-cli dns

# Mail queue management
mailcow-cli queue list
sudo mailcow-cli queue flush
sudo mailcow-cli queue delete ALL

# Backup data
sudo mailcow-cli backup

# Update Mailcow
sudo mailcow-cli update

# Open shell in container
mailcow-cli shell postfix-mailcow
```

Configuration

Access Admin Panel

- **URL:** `https://your-hostname`
- **Username:** `admin`
- **Default Password:** `moohoo`

Change the admin password immediately after first login!

Create Domains and Mailboxes

1. Log into admin panel
2. Go to **Email → Configuration**
3. Add your domain
4. Create mailboxes under the domain

Configure DKIM

1. Go to **Configuration → Configuration & Details**
2. Find **ARC/DKIM keys** section
3. Copy the DKIM record shown

4. Add to your DNS

Configure Email Clients

IMAP Settings:

- Server: `mail.example.com`
- Port: 993 (SSL/TLS) or 143 (STARTTLS)
- Username: Full email address
- Password: Mailbox password

SMTP Settings:

- Server: `mail.example.com`
- Port: 587 (STARTTLS) or 465 (SSL/TLS)
- Username: Full email address
- Password: Mailbox password

Data Storage

EBS Volume Layout

When using a separate EBS volume (recommended):

Path	Contents
`/mnt/mailcow-data/vmail`	Email storage
`/mnt/mailcow-data/mysql`	Database
`/mnt/mailcow-data/redis`	Cache
`/mnt/mailcow-data/rspamd`	Spam filter data
`/mnt/mailcow-data/crypt`	Encryption keys
`/mnt/mailcow-data/backup`	Backup files

Backup Strategy

Automated Backups:

```
# Create backup
sudo mailcow-cli backup
```

```
# Backups stored in:
/mnt/mailcow-data/backup/
```

Manual Backup:

```
# Stop services for consistent backup
cd /opt/mailcow/mailcow-dockerized
sudo docker compose stop

# Backup data volume
sudo tar -czf mailcow-backup-$(date +%Y%m%d).tar.gz /mnt/mailcow-data

# Restart services
sudo docker compose up -d
```

Restore from Backup:

```
sudo mailcow-cli restore /path/to/backup.tar.gz
```

Monitoring

Service Health

```
# Overall status
mailcow-cli status

# Docker container status
docker compose -f /opt/mailcow/mailcow-dockerized/docker-compose.yml ps
```

Logs

```
# All services
mailcow-cli logs

# Specific services
mailcow-cli logs postfix-mailcow   # SMTP
```

```
mailcow-cli logs dovecot-mailcow   # IMAP/POP3
mailcow-cli logs rspamd-mailcow    # Spam filter
mailcow-cli logs nginx-mailcow     # Web
```

Mail Queue

```
# View queue
mailcow-cli queue list

# Flush stuck messages
sudo mailcow-cli queue flush
```

Troubleshooting

Cannot Send Email (Port 25 Blocked)

This is the #1 issue on AWS. See `/opt/mailcow/scripts/AWS-PORT25-GUIDE.md`.

```
# Test outbound port 25
telnet gmail-smtp-in.l.google.com 25
# If it times out, port 25 is blocked
```

SSL Certificate Issues

```
# Check certificate status
docker compose exec acme-mailcow /root/acme.sh --list

# Force certificate renewal
docker compose exec acme-mailcow /root/acme.sh --renew -d mail.example.com --force
```

Cannot Receive Email

1. Check DNS MX record: `dig MX example.com`
2. Check port 25 is open inbound in Security Group
3. Check Postfix logs: `mailcow-cli logs postfix-mailcow`

High Memory Usage

Mailcow runs many services. If memory is constrained:

```
# Edit mailcow.conf
nano /opt/mailcow/mailcow-dockerized/mailcow.conf

# Disable optional services
SKIP_CLAMD=y    # Disable antivirus
SKIP_SOLR=y     # Disable full-text search

# Restart
docker compose up -d
```

Container Not Starting

```
# Check container logs
docker logs mailcowdockerized-postfix-mailcow-1

# Check disk space
df -h

# Check Docker status
systemctl status docker
```

Building the AMI

Prerequisites

1. Launch a clean Ubuntu 24.04 instance (t3.medium)
2. Attach a 30GB root volume

Build Steps

```
# 1. Run setup script
sudo /opt/mailcow/scripts/setup-mailcow.sh
```

```
# 2. Install CLI and scripts
sudo cp mailcow-cli /usr/local/bin/
sudo chmod +x /usr/local/bin/mailcow-cli

sudo cp configure-mailcow.sh first-boot.sh /opt/mailcow/scripts/
sudo chmod +x /opt/mailcow/scripts/*.sh

sudo cp mailcow-firstboot.service /etc/systemd/system/
sudo systemctl enable mailcow-firstboot.service

sudo cp AWS-PORT25-GUIDE.md /opt/mailcow/scripts/

# 3. Test configuration (optional)
sudo /opt/mailcow/scripts/configure-mailcow.sh

# 4. Cleanup for AMI
sudo /opt/mailcow/scripts/cleanup-for-ami.sh

# 5. Create AMI from AWS Console
```

Security Considerations

1. **Change admin password** immediately after configuration
2. **Use Elastic IP** for consistent email reputation
3. **Request reverse DNS (PTR)** from AWS
4. **Keep Mailcow updated** with `mailcow-cli update`
5. **Monitor logs** for suspicious activity
6. **Enable 2FA** in Mailcow admin panel
7. **Backup regularly** to protect your data

Support

Logs Location

- First boot: `/var/log/mailcow/first-boot.log`

- Mailcow: `mailcow-cli logs`
- System: `/var/log/syslog`

Configuration Files

- Mailcow config: `/opt/mailcow/mailcow-dockerized/mailcow.conf`
- Saved info: `/opt/mailcow/config-info.txt`
- Credentials: `/opt/mailcow/.credentials` (root only)

Resources

- **Mailcow Documentation:** https://docs.mailcow.email/
- **Mailcow GitHub:** https://github.com/mailcow/mailcow-dockerized
- **AWS Port 25 Form:** https://aws.amazon.com/forms/ec2-email-limit-rdns-request

---

Version Information

- **Mailcow:** Latest from GitHub (pulled during configuration)
- **Ubuntu:** 24.04 LTS
- **Docker:** Latest stable
- **AMI Version:** 1.0.0

License

Mailcow is licensed under the GNU General Public License v3.0.

This AMI packaging is provided for AWS Marketplace deployment.

Revision #2
Created 2026-01-19 18:46:48 UTC by Admin
Updated 2026-01-19 19:35:16 UTC by Admin