Jitsi Meet

Jitsi Meet: Production ready, open source video conferencing server with Docker, automatic HTTPS via Let's Encrypt, and easy management CLI. No account required for users.

- 1.1.0 - ami-01f0bfb7cb4e60c5e

1.1.0 - ami-01f0bfb7cb4e60c5e

Jitsi Meet - AWS Marketplace AMI

Production-ready Jitsi Meet video conferencing server for AWS, featuring Docker-based deployment with automatic HTTPS via Let's Encrypt.

Overview

Jitsi Meet is a free, open-source video conferencing solution that provides:

- **Secure Video Calls**: End-to-end encrypted video conferencing
- **No Account Required**: Anyone can join meetings with just a link
- **Screen Sharing**: Share your screen with participants
- **Chat**: Built-in text chat during meetings
- **Recording**: Optional recording capabilities (Jibri)
- **Mobile Support**: Native apps for iOS and Android

System Requirements

Resource	Minimum	Recommended
Instance Type	t3.small	t3.medium or larger
vCPUs	2	4+
RAM	4 GB	8 GB+
Root Volume	20 GB	30 GB
Data Volume	20 GB	50 GB+ (for recordings)

Scaling Guidelines

Concurrent Participants	Recommended Instance

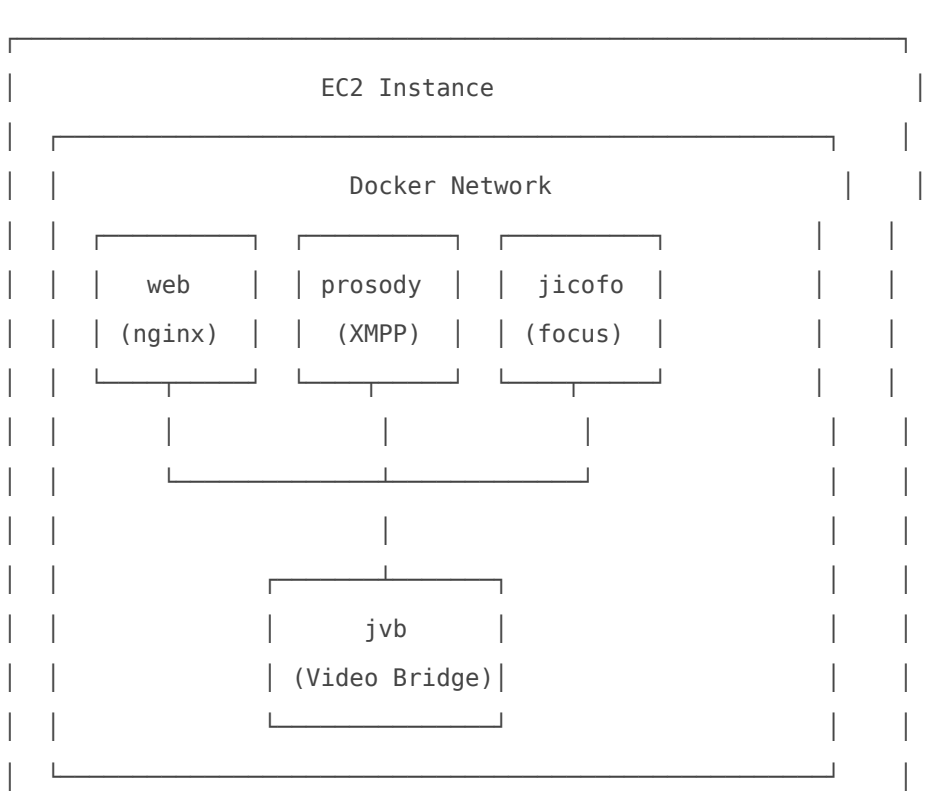
Up to 10	t3.small
10-25	t3.medium
25-50	t3.large
50-100	t3.xlarge
100+	Consider load balancing

Required Ports

Port	Protocol	Direction	Description
22	TCP	Inbound	SSH access
80	TCP	Inbound	HTTP (redirect to HTTPS)
443	TCP	Inbound	HTTPS (Web UI)
10000	UDP	Inbound	WebRTC media (JVB)

Important: Port 10000/UDP is essential for video/audio transmission. Ensure your security group allows this port.

Architecture



```
|                                                          |
|  TCP 80,443 ←→ Web UI                                    |
|  UDP 10000  ←→ Media Streams                             |
└──────────────────────────────────────────────────────────┘
```

Customer Usage

Option 1: Interactive Configuration

After launching an instance from the AMI:

```
ssh -i your-key.pem ubuntu@<instance-ip>
sudo /opt/jitsi/configure-jitsi.sh
```

Follow the prompts to configure:

- Domain name or IP address
- HTTPS with Let's Encrypt (requires domain)
- Email for Let's Encrypt notifications

Option 2: Automated Configuration (User Data)

Provide JSON configuration in EC2 user-data:

```
{
    "host": "jitsi.example.com",
    "enable_https": true,
    "letsencrypt_email": "admin@example.com",
    "timezone": "UTC"
}
```

User Data Parameters

Parameter	Required	Description
`host`	No	Domain name (defaults to public IP)
`enable_https`	No	Enable Let's Encrypt (default: false)
`letsencrypt_email`	If HTTPS	Email for certificate notifications
`timezone`	No	Server timezone (default: UTC)

Option 3: Using IP Address Only

If you don't have a domain name, Jitsi will work with your IP address using a self-signed certificate. Users will see a browser warning which they can accept.

Management Commands

The `jitsi-cli` utility provides easy management:

```
# Check service status
jitsi-cli status

# View logs
jitsi-cli logs           # All services
jitsi-cli logs jvb       # Video Bridge only
jitsi-cli logs web       # Web server only

# Service control (requires sudo)
sudo jitsi-cli start
sudo jitsi-cli stop
sudo jitsi-cli restart

# Update to latest images
sudo jitsi-cli update

# Create backup
sudo jitsi-cli backup

# Show configuration info
jitsi-cli info

# Open shell in container
jitsi-cli shell web

# Show active participants
jitsi-cli participants
```

HTTPS Setup

With Let's Encrypt (Recommended)

Requirements:

- A domain name pointing to your instance IP
- Ports 80 and 443 open
- Valid email address

The setup automatically:

1. Obtains SSL certificate from Let's Encrypt
2. Configures automatic renewal
3. Redirects HTTP to HTTPS

Without Domain (Self-Signed)

When using an IP address:

- Jitsi generates a self-signed certificate
- Users will see a browser security warning
- WebRTC still works after accepting the warning

Bring Your Own Certificate

To use your own certificate:

1. Place files in `/mnt/jitsi-data/certs/`:
   - `cert.crt` - Certificate file
   - `cert.key` - Private key
2. Update docker-compose.yml to mount certificates

Data Storage

Path	Description
`/opt/jitsi`	Configuration and scripts
`/mnt/jitsi-data`	Persistent data
`/mnt/jitsi-data/web`	Web server config

Path	Description
`/mnt/jitsi-data/prosody`	XMPP server data
`/mnt/jitsi-data/jicofo`	Focus component config
`/mnt/jitsi-data/jvb`	Video Bridge config
`/mnt/jitsi-data/certs`	SSL certificates

EBS Volume

If you attach an additional EBS volume, it will be automatically:

- Detected and formatted (if needed)
- Mounted at `/mnt/jitsi-data`
- Added to `/etc/fstab` for persistence

Supported device names:

- `/dev/xvdb`, `/dev/xvdf`
- `/dev/nvme1n1`, `/dev/nvme2n1`
- `/dev/sdf`

Configuration Files

Environment Variables

Located at `/opt/jitsi/.env`:

```
# Public URL (required for production)
PUBLIC_URL=https://jitsi.example.com

# Let's Encrypt
ENABLE_LETSENCRYPT=true
LETSENCRYPT_DOMAIN=jitsi.example.com
LETSENCRYPT_EMAIL=admin@example.com

# Video Bridge
JVB_PORT=10000
JVB_ADVERTISE_IPS=<public-ip>
```

```
# Timezone
TZ=UTC
```

Docker Compose

Located at `/opt/jitsi/docker-compose.yml`

Troubleshooting

No Audio/Video

1. Check UDP port 10000 is open in security group
2. Verify JVB_ADVERTISE_IPS is set to public IP
3. Check logs: `jitsi-cli logs jvb`

SSL Certificate Issues

1. Verify domain points to instance IP
2. Check ports 80/443 are open
3. View acme logs: `jitsi-cli logs acme`
4. Wait a few minutes for certificate issuance

Cannot Connect

1. Check all containers are running: `jitsi-cli status`
2. Verify firewall rules: `sudo ufw status`
3. Test ports: `nc -zv <ip> 443`

Container Restart Loop

1. Check logs for errors: `jitsi-cli logs`
2. Verify environment file: `cat /opt/jitsi/.env`
3. Reset configuration:

```
sudo systemctl stop jitsi
sudo rm -rf /mnt/jitsi-data/*
sudo rm /opt/jitsi/.configured
sudo /opt/jitsi/configure-jitsi.sh
```

Advanced Configuration

Enable Authentication

Edit `/opt/jitsi/.env` and add:

```
ENABLE_AUTH=1
AUTH_TYPE=internal
```

Then restart and create users:

```
sudo jitsi-cli restart
docker exec jitsi-prosody prosodyctl --config /config/prosody.cfg.lua register <username>
meet.jitsi <password>
```

Enable Recording (Jibri)

Recording requires additional setup and resources. See the official Jitsi documentation for Jibri configuration.

Scaling with Multiple JVBs

For large deployments, you can add multiple Video Bridge instances. This requires:

1. Additional EC2 instances for JVBs
2. Configured OCAR (Ocrasia) for load balancing
3. Shared Prosody configuration

Security Best Practices

1. **Keep Updated**: Regularly run `sudo jitsi-cli update`
2. **Use HTTPS**: Always use a domain with Let's Encrypt
3. **Restrict SSH**: Use key-based authentication, disable root login
4. **Monitor Logs**: Check for suspicious activity
5. **Backup Regularly**: Use `jitsi-cli backup`

Support

Logs Location

- First boot: `/var/log/jitsi-firstboot.log`
- Docker logs: `jitsi-cli logs`
- System logs: `journalctl -u jitsi`

Useful Commands

```
# Check Docker status
docker ps

# Check disk space
df -h

# Check memory usage
free -m

# View real-time stats
docker stats
```

License

Jitsi Meet is open-source software licensed under the Apache License 2.0.

Links

- Jitsi Meet Official
- Docker Jitsi Meet
- Jitsi Handbook