Azure Cache for Redis Setup Guide

🚀 Quick Setup in Azure Portal

1. Create Azure Cache for Redis

1. Go to Azure Portal
2. Click **"+ Create a resource"**
3. Search for **"Azure Cache for Redis"**
4. Click **"Create"**

2. Configure Your Cache

Basic Settings:

- **Subscription**: Select your subscription
- **Resource group**: Create new or use existing
- **DNS name**: `your-app-redis` (will be `your-app-redis.redis.cache.windows.net`)
- **Location**: Same region as your app for low latency
- **Cache SKU**:
  - **Development**: Basic C0 (250 MB) - ~$0.02/hour
  - **Production**: Standard C1 (1 GB) or Premium P1 (6 GB)
- **Redis version**: 6 (latest stable)

Networking:

- **Public endpoint**: Yes (for development)
- **Private endpoint**: Consider for production

Advanced:

- **Enable non-SSL port**: NO (keep it secure)
- **Redis Modules**: Not needed for this app
- **Clustering**: No (unless high scale needed)

3. Get Connection Details

After deployment (5-10 minutes):

1. Go to your Redis resource

2. Click **"Access keys"** in left menu
3. Copy:
   - **Host name**: `your-app-redis.redis.cache.windows.net`
   - **Primary key**: Your password
   - **Port**: 6380 (SSL)

🔧 Configure Your Application

Update Your .env File

```bash
# Azure Cache for Redis Configuration
# Option 1: Using connection string (RECOMMENDED)
REDIS_URL=rediss://default:YOUR_PRIMARY_KEY@your-app-redis.redis.cache.windows.net:6380/0

# Option 2: Using individual settings
# REDIS_HOST=your-app-redis.redis.cache.windows.net
# REDIS_PORT=6380
# REDIS_DB=0
# REDIS_PASSWORD=YOUR_PRIMARY_KEY
# REDIS_SSL=true
# REDIS_SSL_CERT_REQS=required

# Rest of your config remains the same...
```

Important Notes:

- Use `rediss://` (with double 's') for SSL connection
- Default username is `default` for Redis 6+
- Port `6380` is the SSL port (required)
- Database is typically `0` (Azure Cache supports 0-15)

🔒 Security Best Practices

1. Use Azure Key Vault (Recommended)

Instead of storing the Redis password in .env:

```python
```

```python
# Add to your config.py
from azure.keyvault.secrets import SecretClient
from azure.identity import DefaultAzureCredential

def get_redis_password_from_keyvault():
    try:
        credential = DefaultAzureCredential()
        client = SecretClient(
            vault_url="https://your-keyvault.vault.azure.net/",
            credential=credential
        )
        secret = client.get_secret("redis-password")
        return secret.value
    except Exception:
        # Fallback to environment variable
        return os.getenv("REDIS_PASSWORD")
```

2. Use Managed Identity (Production)

For Azure App Service or Container Instances:

```python
# The DefaultAzureCredential will automatically use Managed Identity
credential = DefaultAzureCredential()
```

3. IP Restrictions

In Azure Portal → Your Redis → Firewall:

- Add your development machine's IP
- Add your Azure App Service outbound IPs
- Consider using Private Endpoints for production

📊 Monitoring & Diagnostics

Enable Diagnostics

1. Go to your Redis resource
2. Click **"Diagnostic settings"**
3. Add diagnostic setting:
    - Send to Log Analytics workspace

- Enable: AllMetrics, ConnectedClientList

Key Metrics to Monitor

- **Cache hits/misses**: Optimize your caching strategy
- **Connected clients**: Monitor connection leaks
- **Server load**: Upgrade if consistently > 80%
- **Network bandwidth**: Check if hitting limits

Azure CLI Commands

```bash
# Get connection string
az redis show --name your-app-redis --resource-group your-rg --query "hostName" -o tsv

# List access keys
az redis list-keys --name your-app-redis --resource-group your-rg

# Force reboot (if needed)
az redis force-reboot --name your-app-redis --resource-group your-rg --reboot-type AllNodes
```

🧪 Testing Your Connection

1. Test with Redis CLI

```bash
```

```
# Install redis-cli if needed
# macOS: brew install redis
# Ubuntu: sudo apt install redis-tools

# Connect to Azure Cache
redis-cli -h your-app-redis.redis.cache.windows.net -p 6380 -a YOUR_PRIMARY_KEY --tls

# Test commands
> ping
PONG
> set test "Hello Azure"
OK
> get test
"Hello Azure"
> del test
(integer) 1
```

2. Python Test Script

```python
import redis

# Test connection
r = redis.Redis(
    host='your-app-redis.redis.cache.windows.net',
    port=6380,
    password='YOUR_PRIMARY_KEY',
    ssl=True,
    ssl_cert_reqs='required',
    decode_responses=True
)

try:
    print(r.ping())  # Should print True
    r.set('test', 'Hello from Python')
    print(r.get('test'))  # Should print "Hello from Python"
    r.delete('test')
    print("✅ Connection successful!")
except Exception as e:
    print(f"× Connection failed: {e}")
```

💰 Cost Optimization

Development

- Use **Basic C0** tier (~$16/month)
- Consider **Azure Free Tier** (includes Basic C0 for 12 months)
- Delete cache when not in use

Production

- Start with **Standard C1** (~$60/month)
- Enable **persistence** only if needed (adds cost)
- Use **clustering** only for high scale
- Monitor metrics to right-size

Auto-scaling (Premium only)

```bash
# Example auto-scale rule
az monitor autoscale create \
  --resource-group your-rg \
  --resource your-app-redis \
  --resource-type Microsoft.Cache/Redis \
  --min-count 1 \
  --max-count 4 \
  --count 1
```

🚨 Common Issues & Solutions

"SSL connection error"

```python
# Make sure you're using SSL settings
REDIS_SSL=true
REDIS_SSL_CERT_REQS=required  # or "none" for testing
```

"Connection timeout"

- Check firewall rules in Azure Portal
- Verify your IP is whitelisted

- Check if using correct port (6380 for SSL)

"Authentication failed"

- Verify you're using the correct access key
- Try regenerating keys in Azure Portal
- Check if key contains special characters (URL encode them)

"Certificate verification failed"

```python
# For development only (not recommended for production)
REDIS_SSL_CERT_REQS=none

# Better solution: Update certificates
# Ubuntu: sudo apt update && sudo apt install ca-certificates
# macOS: brew install ca-certificates
```

🔄 Migration from Local Redis

Export Local Data (if needed)

```bash
# On local Redis
redis-cli --rdb dump.rdb

# Or export specific keys
redis-cli --scan --pattern "refresh_token:*" > tokens.txt
```

Import to Azure Cache

```bash
# Use redis-cli with Azure Cache
while read key; do
  value=$(redis-cli get "$key")
  redis-cli -h your-app.redis.cache.windows.net -p 6380 -a YOUR_KEY --tls set "$key" "$value"
done < tokens.txt
```

📱 Connection String Formats

For Different Environments

```bash
# Development (local Redis)
REDIS_URL=redis://localhost:6379/0

# Azure Cache for Redis
REDIS_URL=rediss://default:YOUR_KEY@your-app.redis.cache.windows.net:6380/0

# With special characters in password (URL encode)
REDIS_URL=rediss://default:abc%40123%23@your-app.redis.cache.windows.net:6380/0

# Redis 6+ with ACL username
REDIS_URL=rediss://myusername:mypassword@your-app.redis.cache.windows.net:6380/0
```

✅ Final Checklist

- [ ] Azure Cache for Redis instance created
- [ ] Firewall rules configured
- [ ] Connection string added to .env
- [ ] Application tested with Azure Cache
- [ ] Monitoring enabled
- [ ] Backup strategy defined (if using persistence)
- [ ] Alerts configured for critical metrics
- [ ] Cost alerts set up

🎉 You're Ready!

Your app is now using enterprise-grade Redis hosting with:

- ✅ High availability (99.9% SLA)
- ✅ Automatic patching
- ✅ Built-in monitoring
- ✅ Encryption at rest and in transit
- ✅ Geo-replication (Premium tier)
- ✅ Backup/restore capabilities