# 🪟 CYPHER Windows Server 2019 Deployment Guide

## 🎯 **Complete Step-by-Step Deployment**

### **Prerequisites**

* Windows Server 2019 EC2 instance (t3.large)
* Your zipped code in S3 bucket
* RDS PostgreSQL database
* Domain name (optional)

## 📋 **Phase 1: EC2 Instance Setup**

### **Step 1: Launch EC2 Instance**

# In AWS Console:  
# 1. Launch Instance → Windows Server 2019 Base  
# 2. Instance Type: t3.large (2 vCPU, 8 GB RAM)  
# 3. Key Pair: Create/Select your key pair  
# 4. Security Group: Create with these rules:  
# - RDP (3389): Your IP only  
# - HTTP (80): 0.0.0.0/0  
# - HTTPS (443): 0.0.0.0/0  
# - Custom TCP (3001): 0.0.0.0/0 (API port)  
# 5. Storage: 30 GB GP3

### **Step 2: Connect to Instance**

# Get Windows password using your key pair  
# Connect via Remote Desktop (RDP)  
# Username: Administrator  
# Password: [Retrieved from AWS Console]

## 🔧 **Phase 2: Software Installation**

### **Step 3: Install IIS and Features**

# Open PowerShell as Administrator  
# Install IIS with required features  
Enable-WindowsOptionalFeature -Online -FeatureName IIS-WebServerRole  
Enable-WindowsOptionalFeature -Online -FeatureName IIS-WebServer  
Enable-WindowsOptionalFeature -Online -FeatureName IIS-CommonHttpFeatures  
Enable-WindowsOptionalFeature -Online -FeatureName IIS-HttpErrors  
Enable-WindowsOptionalFeature -Online -FeatureName IIS-HttpLogging  
Enable-WindowsOptionalFeature -Online -FeatureName IIS-RequestFiltering  
Enable-WindowsOptionalFeature -Online -FeatureName IIS-StaticContent  
Enable-WindowsOptionalFeature -Online -FeatureName IIS-DefaultDocument  
Enable-WindowsOptionalFeature -Online -FeatureName IIS-DirectoryBrowsing  
Enable-WindowsOptionalFeature -Online -FeatureName IIS-ASPNET45  
  
Write-Host "✅ IIS installed successfully!" -ForegroundColor Green

### **Step 4: Install Node.js**

# Download and install Node.js LTS  
$nodeUrl = "https://nodejs.org/dist/v18.19.0/node-v18.19.0-x64.msi"  
$nodeInstaller = "$env:TEMP\nodejs-installer.msi"  
  
Invoke-WebRequest -Uri $nodeUrl -OutFile $nodeInstaller  
Start-Process -FilePath $nodeInstaller -ArgumentList "/quiet" -Wait  
  
# Add Node.js to PATH (restart PowerShell after this)  
$env:PATH += ";C:\Program Files\nodejs"  
  
# Verify installation  
node --version  
npm --version  
  
Write-Host "✅ Node.js installed successfully!" -ForegroundColor Green

### **Step 5: Install IISNode**

# Download and install iisnode  
$iisnodeUrl = "https://github.com/Azure/iisnode/releases/download/v0.2.26/iisnode-full-v0.2.26-x64.msi"  
$iisnodeInstaller = "$env:TEMP\iisnode-installer.msi"  
  
Invoke-WebRequest -Uri $iisnodeUrl -OutFile $iisnodeInstaller  
Start-Process -FilePath $iisnodeInstaller -ArgumentList "/quiet" -Wait  
  
Write-Host "✅ IISNode installed successfully!" -ForegroundColor Green

### **Step 6: Install Git and AWS CLI**

# Install Git  
$gitUrl = "https://github.com/git-for-windows/git/releases/download/v2.43.0.windows.1/Git-2.43.0-64-bit.exe"  
$gitInstaller = "$env:TEMP\git-installer.exe"  
  
Invoke-WebRequest -Uri $gitUrl -OutFile $gitInstaller  
Start-Process -FilePath $gitInstaller -ArgumentList "/SILENT" -Wait  
  
# Install AWS CLI  
$awsUrl = "https://awscli.amazonaws.com/AWSCLIV2.msi"  
$awsInstaller = "$env:TEMP\aws-cli-installer.msi"  
  
Invoke-WebRequest -Uri $awsUrl -OutFile $awsInstaller  
Start-Process -FilePath $awsInstaller -ArgumentList "/quiet" -Wait  
  
Write-Host "✅ Git and AWS CLI installed successfully!" -ForegroundColor Green

## 🗄️ **Phase 3: Database Configuration**

### **Step 7: Configure PostgreSQL Connection**

# Create environment configuration  
$appDir = "C:\inetpub\wwwroot\cypher"  
New-Item -ItemType Directory -Force -Path $appDir  
  
# Create .env file with your RDS details  
@"  
NODE\_ENV=production  
PORT=3001  
  
# Database Configuration  
DATABASE\_URL=postgresql://username:password@your-rds-endpoint.amazonaws.com:5432/your-database-name  
DB\_HOST=your-rds-endpoint.amazonaws.com  
DB\_PORT=5432  
DB\_NAME=your-database-name  
DB\_USER=your-username  
DB\_PASSWORD=your-password  
  
# JWT Configuration  
JWT\_SECRET=your-super-secret-jwt-key-here  
JWT\_EXPIRES\_IN=24h  
  
# CORS Configuration  
CORS\_ORIGIN=http://your-domain.com  
  
# Rate Limiting  
RATE\_LIMIT\_WINDOW\_MS=900000  
RATE\_LIMIT\_MAX\_REQUESTS=100  
"@ | Out-File -FilePath "$appDir\.env" -Encoding UTF8  
  
Write-Host "✅ Environment configuration created!" -ForegroundColor Green

## 📦 **Phase 4: Application Deployment**

### **Step 8: Download and Extract Code from S3**

# Configure AWS credentials  
aws configure  
# Enter your AWS Access Key ID, Secret Access Key, Region (us-east-1), and output format (json)  
  
# Download your code from S3  
$s3Bucket = "your-s3-bucket-name"  
$s3Key = "your-code-zip-file.zip"  
$downloadPath = "$env:TEMP\cypher-code.zip"  
  
aws s3 cp "s3://$s3Bucket/$s3Key" $downloadPath  
  
# Extract to application directory  
Expand-Archive -Path $downloadPath -DestinationPath $appDir -Force  
  
Write-Host "✅ Code downloaded and extracted!" -ForegroundColor Green

### **Step 9: Install Dependencies**

# Navigate to API directory and install dependencies  
cd "$appDir\api"  
npm install --production  
  
# Navigate to client directory and build  
cd "$appDir\client"  
npm install  
npm run build  
  
Write-Host "✅ Dependencies installed and client built!" -ForegroundColor Green

## 🌐 **Phase 5: IIS Configuration**

### **Step 10: Configure IIS Site**

# Import IIS module  
Import-Module WebAdministration  
  
# Remove default website  
Remove-Website -Name "Default Web Site" -ErrorAction SilentlyContinue  
  
# Create new application pool  
New-WebAppPool -Name "CypherAppPool" -Force  
Set-ItemProperty -Path "IIS:\AppPools\CypherAppPool" -Name processModel.identityType -Value ApplicationPoolIdentity  
Set-ItemProperty -Path "IIS:\AppPools\CypherAppPool" -Name recycling.periodicRestart.time -Value "00:00:00"  
  
# Create website for client (React app)  
New-Website -Name "CypherClient" -Port 80 -PhysicalPath "$appDir\client\dist" -ApplicationPool "CypherAppPool"  
  
# Create application for API  
New-WebApplication -Site "CypherClient" -Name "api" -PhysicalPath "$appDir\api" -ApplicationPool "CypherAppPool"  
  
Write-Host "✅ IIS sites configured!" -ForegroundColor Green

### **Step 11: Configure IISNode for API**

# Create web.config for API  
$webConfigContent = @"  
<?xml version="1.0" encoding="utf-8"?>  
<configuration>  
 <system.webServer>  
 <handlers>  
 <add name="iisnode" path="server.js" verb="\*" modules="iisnode" />  
 </handlers>  
 <rewrite>  
 <rules>  
 <rule name="NodeInspector" patternSyntax="ECMAScript" stopProcessing="true">  
 <match url="^server.js\/debug[\/]?" />  
 </rule>  
 <rule name="StaticContent">  
 <action type="Rewrite" url="public{REQUEST\_URI}"/>  
 </rule>  
 <rule name="DynamicContent">  
 <conditions>  
 <add input="{REQUEST\_FILENAME}" matchType="IsFile" negate="True"/>  
 </conditions>  
 <action type="Rewrite" url="server.js"/>  
 </rule>  
 </rules>  
 </rewrite>  
 <security>  
 <requestFiltering>  
 <hiddenSegments>  
 <remove segment="bin"/>  
 </hiddenSegments>  
 </requestFiltering>  
 </security>  
 <httpErrors existingResponse="PassThrough" />  
 <iisnode watchedFiles="web.config;\*.js"/>  
 </system.webServer>  
</configuration>  
"@  
  
$webConfigContent | Out-File -FilePath "$appDir\api\web.config" -Encoding UTF8  
  
Write-Host "✅ IISNode configured!" -ForegroundColor Green

## 🔗 **Phase 6: Domain and SSL Setup**

### **Step 12: Configure Domain (Optional)**

# If you have a domain, update DNS to point to your EC2 public IP  
# Then update IIS binding  
Remove-WebBinding -Site "CypherClient" -Port 80 -Protocol http  
New-WebBinding -Site "CypherClient" -Name "your-domain.com" -Port 80 -Protocol http  
  
Write-Host "✅ Domain configured!" -ForegroundColor Green

### **Step 13: Install SSL Certificate (Optional)**

# For production, install SSL certificate  
# You can use AWS Certificate Manager or Let's Encrypt  
# This example shows basic HTTP setup  
Write-Host "ℹ️ For production, configure SSL certificate" -ForegroundColor Yellow

## 🚀 **Phase 7: Final Configuration and Testing**

### **Step 14: Start Services and Test**

# Restart IIS  
iisreset  
  
# Test API endpoint  
Start-Sleep -Seconds 10  
try {  
 $response = Invoke-WebRequest -Uri "http://localhost/api/health" -TimeoutSec 30  
 if ($response.StatusCode -eq 200) {  
 Write-Host "✅ API is responding!" -ForegroundColor Green  
 }  
} catch {  
 Write-Host "⚠️ API health check failed - check logs" -ForegroundColor Yellow  
}  
  
# Test client  
try {  
 $response = Invoke-WebRequest -Uri "http://localhost" -TimeoutSec 30  
 if ($response.StatusCode -eq 200) {  
 Write-Host "✅ Client is responding!" -ForegroundColor Green  
 }  
} catch {  
 Write-Host "⚠️ Client not responding - check configuration" -ForegroundColor Yellow  
}  
  
Write-Host "🎉 Deployment completed!" -ForegroundColor Green  
Write-Host "🌐 Your application should be available at: http://your-ec2-public-ip" -ForegroundColor Cyan

## 📊 **Phase 8: Monitoring and Maintenance**

### **Step 15: Create Monitoring Script**

# Create monitoring script  
$monitorScript = @"  
# CYPHER Application Monitor  
Write-Host "🔍 CYPHER Application Status" -ForegroundColor Cyan  
Write-Host "=========================" -ForegroundColor Cyan  
  
# Check IIS status  
`$iisStatus = Get-Service -Name W3SVC  
Write-Host "IIS Status: `$(`$iisStatus.Status)" -ForegroundColor Yellow  
  
# Check application pool  
`$appPoolStatus = Get-WebAppPoolState -Name "CypherAppPool"  
Write-Host "App Pool Status: `$(`$appPoolStatus.Value)" -ForegroundColor Yellow  
  
# Test endpoints  
try {  
 `$apiResponse = Invoke-WebRequest -Uri "http://localhost/api/health" -TimeoutSec 10  
 Write-Host "✅ API Health: OK (`$(`$apiResponse.StatusCode))" -ForegroundColor Green  
} catch {  
 Write-Host "❌ API Health: Failed" -ForegroundColor Red  
}  
  
try {  
 `$clientResponse = Invoke-WebRequest -Uri "http://localhost" -TimeoutSec 10  
 Write-Host "✅ Client: OK (`$(`$clientResponse.StatusCode))" -ForegroundColor Green  
} catch {  
 Write-Host "❌ Client: Failed" -ForegroundColor Red  
}  
"@  
  
$monitorScript | Out-File -FilePath "C:\Scripts\Monitor-Cypher.ps1" -Encoding UTF8  
New-Item -ItemType Directory -Force -Path "C:\Scripts"  
  
Write-Host "✅ Monitoring script created at C:\Scripts\Monitor-Cypher.ps1" -ForegroundColor Green

## 🎯 **Quick Reference Commands**

# Restart IIS  
iisreset  
  
# Check application status  
C:\Scripts\Monitor-Cypher.ps1  
  
# View IIS logs  
Get-Content "C:\inetpub\logs\LogFiles\W3SVC1\\*.log" | Select-Object -Last 50  
  
# Restart application pool  
Restart-WebAppPool -Name "CypherAppPool"  
  
# Check Windows Event Logs  
Get-EventLog -LogName Application -Source "IIS\*" -Newest 10

## 🔧 **Troubleshooting**

### **Common Issues:**

1. **API not responding**: Check web.config and ensure server.js exists
2. **Database connection failed**: Verify RDS security group allows EC2 access
3. **Client not loading**: Ensure build files are in correct directory
4. **Port conflicts**: Check if port 3001 is available

### **Log Locations:**

* IIS Logs: C:\inetpub\logs\LogFiles\W3SVC1\
* Application Logs: Windows Event Viewer → Application
* IISNode Logs: C:\inetpub\wwwroot\cypher\api\

**🎉 Your CYPHER application should now be running on Windows Server 2019 with IIS!**