Lab 9: Hosting i Wdrażanie

Deployment aplikacji . NET

Materiały na kolokwium - Część 3

7 grudnia 2025

Spis treści

1 Wprowadzenie do Hostingu

1.1 Typy Hostingu

Klasyfikacja Hostingu Aplikacji

1. **Hosting Lokalny (On-Premises):**
   - Własny serwer fizyczny lub lokalny komputer
   - Pełna kontrola nad infrastrukturą
   - IIS (Windows) / Kestrel / Docker
   - Odpowiedzialność za maintenance, updates, security
2. **VPS (Virtual Private Server):**
   - Wirtualna maszyna w chmurze
   - Root access, pełna kontrola OS
   - Providers: Azure VM, Oracle Cloud, Google Cloud, DigitalOcean
   - Ręczna konfiguracja i zarządzanie
3. **Managed Hosting (PaaS):**
   - Platform as a Service
   - Azure App Service, Heroku, Vercel
   - Zero infrastructure management
   - Auto-scaling, monitoring, backups
   - Wyższy koszt, mniejsza kontrola
4. **Serverless:**
   - Azure Functions, AWS Lambda
   - Pay-per-execution
   - Automatic scaling
   - Ograniczenia (timeout, stateless)

1.2 Kluczowe Pojęcia

Terminologia Hostingu

Reverse Proxy:

- Serwer pośredniczący między internetem a aplikacją
- Nginx, IIS, Apache
- Funkcje: SSL termination, load balancing, caching
- Przykład: Nginx (port 80/443) → Kestrel (port 5000)

Load Balancing:

- Dystrybucja ruchu między wiele instancji aplikacji
- Round-robin, least connections, IP hash
- High availability i performance

SSL/TLS:

- Encryption komunikacji (HTTPS)
- Certyfikaty: Let's Encrypt (free), Cloudflare, commercial
- Port 443 (HTTPS) vs 80 (HTTP)

Public IP:

- Adres IP widoczny z internetu
- Wymagany do bezpośredniego dostępu z zewnątrz
- Alternatywa: tunelowanie (Cloudflare Tunnel)

DNS (Domain Name System):

- Mapowanie domeny na IP address
- Przykład: myapp.com → 20.123.45.67
- Rekordy: A, CNAME, TXT, MX

Port Forwarding:

- Przekierowanie ruchu z routera do lokalnej maszyny
- Konfiguracja w routerze domowym
- External port → Internal IP:port
- Przykład: 80 (external) → 192.168.1.10:5000 (internal)

2 Hosting na Własnym Serwerze

2.1 Kestrel (ASP.NET Core)

Kestrel Web Server

Czym jest Kestrel:

- Built-in web server w ASP. NET Core
- Cross-platform (Windows, Linux, macOS)
- High performance (async I/O)
- Default server dla . NET applications

Uruchomienie lokalnie:

- `dotnet run` - development mode
- `dotnet myapp. dll` - production
- Default: `http://localhost:5000`
- Konfiguracja portów: `appsettings.json` lub environment variables

Expose na sieć lokalną:

- Bind do `0.0.0.0` zamiast `localhost`
- `-urls "http://0.0.0.0:5000"`
- Dostęp z innych komputerów: `http://<IP>:5000`
- Firewall: otwórz port 5000

Production considerations:

- Kestrel alone = OK dla internal apps
- Za reverse proxy (Nginx/IIS) = recommended dla public
- Reason: security, SSL termination, load balancing

2.2 IIS (Internet Information Services)

IIS - Windows Web Server

Instalacja:

- Windows Features → IIS (Internet Information Services)
- Zainstaluj ASP. NET Core Hosting Bundle
- Hosting Bundle = . NET Runtime + IIS integration module

Konfiguracja witryny:

1. IIS Manager → Sites → Add Website
2. Site name: MyApp
3. Physical path: `C:\inetpub\myapp`
4. Binding:
    - Type: http/https
    - Port: 80 (HTTP) lub 443 (HTTPS)
    - Host name: myapp.com (optional)
5. Application Pool: .NET CLR Version = "No Managed Code"

Deployment:

- Publish z Visual Studio → Folder
- Copy plików do physical path
- Restart Application Pool

SSL Certificate:

- Let's Encrypt - darmowy SSL (90 dni ważności)
- Certbot - automatic renewal
- IIS Manager → Server Certificates → Import
- Binding → Add https (port 443) z certificate

Firewall i Port Forwarding

Windows Firewall:

- Inbound Rules → New Rule
- Port: 80 (HTTP), 443 (HTTPS)
- Allow connection
- Apply to Domain, Private, Public (wybierz odpowiednie)

Router Port Forwarding:

- Panel administracyjny routera (np. 192.168.1. 1)
- Port Forwarding / Virtual Server section
- External port: 80 → Internal IP: 192.168.1.10, port 80
- External port: 443 → Internal IP: 192.168.1. 10, port 443
- Save i restart router

Test:

- Z internetu: `http://<public-IP>`
- Lub `http://myapp.com` (jeśli DNS skonfigurowane)

2.3 Utrzymanie i Monitoring

Maintenance Aplikacji

Restarty:

- IIS: Recycle Application Pool
- Kestrel: restart procesu (`systemctl restart myapp`)
- Windows Service: Service Manager → Restart

Updates:

- Deploy nowej wersji (publish)
- Stop application
- Replace plików
- Start application
- Zero-downtime: deployment slots (Azure) lub blue-green deployment

Logi:

- ASP. NET Core: logging do pliku (`Serilog, NLog`)
- IIS: Event Viewer → Application logs
- Location: `C:\inetpub\logs` (IIS)
- Rotation: automatic (old logs archived/deleted)

Monitoring:

- Health checks endpoint (`/health`)
- Uptime monitoring (UptimeRobot, Pingdom - external)
- Application Insights (Azure)
- CPU/Memory monitoring (Task Manager, Performance Monitor)

3 Maszyny Wirtualne i Proxmox VE

3.1 Proxmox VE

Proxmox Virtual Environment

Czym jest Proxmox:

- Open-source hypervisor (virtualization platform)
- Bazuje na KVM (Kernel Virtual Machine) + LXC (Linux Containers)
- Web-based management interface
- Free (community edition)

Use case:

- Tworzenie wielu VM na jednym fizycznym serwerze
- Izolacja aplikacji (każda w osobnym VM)
- Testing environments (snapshots, clones)
- Home lab / lab developerski

Zalety:

- (+) Free i open-source
- (+) Web GUI (easy management)
- (+) Snapshots i backups
- (+) High performance

3.2 Tworzenie VM

Workflow Tworzenia VM

Krok 1: Utwórz VM

- Proxmox Web UI → Create VM
- OS: Windows Server lub Ubuntu Server
- Resources: CPU (2-4 cores), RAM (2-4 GB), Disk (20-50 GB)
- Network: Bridge mode (recommended dla external access)

Krok 2: Instalacja OS

- Mount ISO (Windows Server / Ubuntu Server)
- Start VM i instaluj OS
- Konfiguracja sieci (static IP lub DHCP)

Krok 3: Instalacja . NET

- Ubuntu: `sudo apt install dotnet-sdk-9.0`
- Windows: Download .NET SDK installer
- Verify: `dotnet -version`

Krok 4: Deploy Aplikacji

- SCP (Secure Copy): `scp -r ./publish user@vm-ip:/var/www/myapp`
- Git clone: `git clone <repo>; dotnet publish`
- Run: `dotnet myapp. dll`

3.3 Networking w Proxmox

Networking Modes

NAT (Network Address Translation):

- VM za routerem Proxmox
- Private IP (np. 10.0.0.x)
- Port forwarding z host do VM
- Use case: internal testing, nie potrzebujesz external access

Bridge:

- VM ma własny IP w sieci (jak fizyczna maszyna)
- Direct access z sieci lokalnej
- Możliwy external access (z port forwarding na routerze)
- Use case: production-like environment

VLAN:

- Izolowana sieć wirtualna
- Segmentacja (np. production vs development)
- Advanced use case

3.4 Snapshots i Backups

Data Protection

Snapshots:

- Point-in-time kopia stanu VM
- Instant (seconds)
- Use case: przed update/upgrade (quick rollback)
- Limit: nie zastępuje backups (snapshots mogą się uszkodzić)

Backups:

- Full backup VM (disk image)
- Scheduled backups (daily, weekly)
- Storage: external disk, NAS, cloud (S3, Azure Blob)
- Restore: pełny restore VM z backupu

Best practice:

- Snapshot przed każdą zmianą
- Daily backups (retention: 7 dni)
- Weekly backups (retention: 4 tygodnie)
- Test restore (verify backups działają)

4 Cloudflare Tunnel

4.1 Problem i Rozwiązanie

Ekspozycja Aplikacji Lokalnej

Problem:

- Aplikacja działa lokalnie (localhost:5000)
- Brak publicznego IP
- Router za CGNAT (carrier-grade NAT) - nie można port forward
- Chcesz udostępnić aplikację z internetu

Tradycyjne rozwiązania:

- Port forwarding (wymaga public IP)
- VPN (kompleksowe, wymaga konfiguracji klienta)
- Ngrok (limited free tier)

Cloudflare Tunnel (Zero Trust):

- Bezpieczne tunelowanie bez public IP
- Outbound connection (firewall-friendly)
- Cloudflare jako reverse proxy
- Automatic SSL
- DDoS protection
- FREE tier dostępny

4.2 Jak Działa Cloudflare Tunnel

Architektura

Flow:

1. `cloudflared` daemon na lokalnym serwerze
2. Outbound encrypted connection do Cloudflare edge (port 443)
3. Cloudflare publishes domain (myapp. com)
4. User request: Internet → Cloudflare edge
5. Cloudflare forward przez tunnel → local app (localhost:5000)
6. Response: local app → Cloudflare → User

Kluczowe właściwości:

- **No inbound ports** - zero port forwarding needed
- **Firewall-friendly** - tylko outbound HTTPS (443)
- **Encrypted** - end-to-end TLS
- **Global network** - Cloudflare CDN (low latency)

4.3 Konfiguracja

Setup Cloudflare Tunnel

Krok 1: Przygotowanie

- Załóż konto Cloudflare (free)
- Dodaj domenę do Cloudflare (zmień nameservers)
- Wait for DNS propagation (24h, często szybciej)

Krok 2: Instalacja cloudflared

- Download: cloudflare.com/products/tunnel
- Windows: installer exe
- Linux: `wget <url>; dpkg -i cloudflared. deb`
- Verify: `cloudflared -version`

Krok 3: Utworzenie tunelu

- Login: `cloudflared tunnel login` (otwiera browser, authorize)
- Create tunnel: `cloudflared tunnel create myapp`
- Note tunnel ID (UUID)
- Credentials saved: `/.cloudflared/<UUID>. json`

Krok 4: Konfiguracja routingu

- Mapowanie: tunnel → local service
- `cloudflared tunnel route dns myapp myapp.com`
- To tworzy DNS record: `myapp.com` → `<UUID>.cfargotunnel.com`

Krok 5: Config file

- `/.cloudflared/config.yml`
- Specify: tunnel ID, credentials, ingress rules
- Ingress: `hostname: myapp. com → service: http://localhost:5000`

Krok 6: Uruchomienie

- `cloudflared tunnel run myapp`
- Daemon w background (systemd service on Linux)
- Auto-start on boot

Krok 7: Test

- Z internetu: `https://myapp.com`
- Automatic HTTPS (Cloudflare certificate)
- No SSL setup needed on local server

4.4 Zalety Cloudflare Tunnel

Benefits

Security:

- No exposed ports (brak attack surface)
- DDoS protection (Cloudflare absorbs attacks)
- Web Application Firewall (WAF) - optional
- Rate limiting - built-in

SSL/TLS:

- Automatic SSL certificates
- No Let's Encrypt setup needed
- Always HTTPS
- Auto-renewal (Cloudflare managed)

Performance:

- Global CDN (200+ locations)
- Low latency dla użytkowników worldwide
- Caching static content

Cost:

- FREE tier (unlimited tunnels)
- No bandwidth charges
- No per-request fees

5 VPS Hosting

5.1 Czym jest VPS

Virtual Private Server

Definicja:

- Wirtualna maszyna w chmurze (datacenter)
- Dedicated resources (CPU, RAM, disk)
- Root/admin access
- Public IP address
- 24/7 uptime (SLA 99.9%+)

Vs Shared Hosting:

- VPS = dedicated resources, full control
- Shared = shared resources, limited control, cheaper

Vs Dedicated Server:

- VPS = virtual, cheaper, easier scaling
- Dedicated = physical machine, max performance, expensive

5.2 Darmowe Opcje VPS

Free Tier Providers

Oracle Cloud Free Tier:

- **FREE forever** (nie trial, permanent free)
- 2x VM: 1 OCPU, 1 GB RAM each (AMD/Intel)
- LUB: 4x ARM cores, 24 GB RAM (Ampere A1)
- 200 GB Block Storage
- 10 TB outbound bandwidth/month
- Public IP included
- Use case: production workloads (stable, reliable)

Google Cloud Free Tier:

- 1x e2-micro VM (0.25-1 vCPU, 1 GB RAM)
- FREE forever
- 30 GB storage
- 1 GB outbound bandwidth/month (USA only)
- Use case: small apps, testing

Uwaga:

- Wymagana karta kredytowa (weryfikacja, nie charging)
- Oracle: może wymagać waiting list (popular)
- Free tier limits - monitor usage

5.3 Konfiguracja VPS

Setup Ubuntu VPS

Krok 1: Create VM

- Oracle/Google Cloud Console → Compute → Create Instance
- OS: Ubuntu 22.04/24.04 LTS
- Region: wybierz closest do target users
- SSH key: upload public key (lub create new)

Krok 2: SSH Access

- `ssh ubuntu@<public-IP>` (Linux/Mac)
- PuTTY (Windows) z private key
- First login: change password (security)

Krok 3: System Updates

- `sudo apt update`
- `sudo apt upgrade -y`
- `sudo reboot` (jeśli kernel update)

Krok 4: Instalacja . NET

- `wget https://dot.net/v1/dotnet-install.sh`
- `bash dotnet-install.sh -channel 9. 0`
- Add to PATH: `export PATH=$PATH:$HOME/.dotnet`
- Verify: `dotnet -version`

Krok 5: Firewall

- `sudo ufw allow 22` (SSH)
- `sudo ufw allow 80` (HTTP)
- `sudo ufw allow 443` (HTTPS)
- `sudo ufw enable`

5.4 Deploy Aplikacji

Deployment Methods

1. Git Clone

- `git clone https://github.com/user/myapp.git`
- `cd myapp`
- `dotnet publish -c Release -o ./publish`
- `cd publish`
- `dotnet myapp. dll`

2. SCP (Secure Copy)

- Local: `dotnet publish -c Release`
- `scp -r ./bin/Release/net9.0/publish/* user@vps:/var/www/myapp`
- SSH to VPS: `cd /var/www/myapp; dotnet myapp. dll`

3. GitHub Actions (CI/CD)

- Workflow: build → publish → deploy to VPS via SSH
- Automatic deployment on push to main branch
- Secrets: VPS IP, SSH key, username

5.5 Nginx jako Reverse Proxy

Nginx Configuration

Instalacja:

- `sudo apt install nginx`
- `sudo systemctl start nginx`
- `sudo systemctl enable nginx` (auto-start on boot)

Konfiguracja:

- `sudo nano /etc/nginx/sites-available/myapp`
- Server block:
  - `listen 80;`
  - `server_name myapp.com;`
  - `location / { proxy_pass http://localhost:5000; }`
  - Headers: `proxy_set_header Host $host;`
- Symlink: `sudo ln -s /etc/nginx/sites-available/myapp /etc/nginx/sites-enabled/`
- Test: `sudo nginx -t`
- Reload: `sudo systemctl reload nginx`

SSL (Let's Encrypt):

- `sudo apt install certbot python3-certbot-nginx`
- `sudo certbot -nginx -d myapp.com`
- Automatic: certificate + nginx config update (port 443)
- Auto-renewal: `sudo certbot renew -dry-run` (test)
- Cron job: renewal check 2x daily (automatic)

Workflow:

- User → Nginx (port 80/443) → Kestrel (port 5000)
- Nginx handles: SSL, static files, load balancing
- Kestrel handles: application logic

6 Azure App Service

6.1 Podstawy

Azure App Service - PaaS

Czym jest:

- Managed platform do hostingu web apps
- Zero infrastructure management
- Auto-scaling, load balancing
- Built-in CI/CD (GitHub, Azure DevOps)
- Deployment slots (staging, production)

Wspierane:

- . NET, Node.js, Python, Java, PHP, Ruby
- Docker containers
- Static sites

6.2 Pricing Tiers

App Service Plans

Free (F1):

- Shared infrastructure
- 1 GB disk, 1 GB RAM
- 60 min CPU/day limit
- No custom domain, no SSL
- Use case: development, testing

Basic (B1):

- $13/month (54 PLN)
- Dedicated VM (1 core, 1. 75 GB RAM)
- Always On support
- Custom domain + SSL
- Manual scaling (up to 3 instances)
- Use case: small production apps

Standard (S1):

- $70/month
- Auto-scaling (up to 10 instances)
- Deployment slots (5)
- Daily backups
- Use case: production apps z traffic

Premium (P1V2):

- $150/month
- Faster scaling, better hardware
- More slots (20)
- VNET integration
- Use case: enterprise, high-traffic

6.3 Deployment

Deploy Methods

1. Visual Studio

- Right-click project → Publish
- Target: Azure App Service
- Select subscription, resource group, plan
- One-click deploy

2. Azure CLI

- `az webapp up -name myapp -resource-group rg -runtime "DOTNETCORE:9.0"`
- Single command: create + deploy

3. GitHub Actions

- Workflow file: `. github/workflows/deploy.yml`
- Trigger: push to main
- Steps: checkout → build → publish → deploy
- Secrets: Azure credentials (service principal)

4. Docker Container

- Build image: `docker build -t myapp .`
- Push to ACR (Azure Container Registry)
- App Service pull from ACR
- Container Apps (alternative) - serverless containers

6.4 Configuration

App Settings

Application Settings:

- Environment variables dla aplikacji
- Przykład: `ConnectionStrings:DefaultConnection`
- Override `appsettings. json`
- Azure Portal → App Service → Configuration → Application Settings

Connection Strings:

- Database connection strings
- Secure storage (not in code)
- Types: SQL Server, MySQL, PostgreSQL, Custom

Key Vault Integration:

- Reference secrets z Azure Key Vault
- Format: `@Microsoft. KeyVault(SecretUri=https://... )`
- Managed Identity (no credentials in config)
- Best practice dla production secrets

6.5 Monitoring

Application Insights

Telemetria:

- Request rates, response times, failure rates
- Dependency calls (database, APIs)
- Exceptions i stack traces
- Custom events i metrics

Setup:

- Enable Application Insights w App Service
- Instrumentation Key automatically injected
- Zero code changes (auto-instrumentation)

Features:

- Live metrics - real-time dashboard
- Application Map - dependencies visualization
- Failures - exception analysis
- Performance - slow requests
- Alerts - email/SMS notifications

7 Domeny i DNS

7.1 Konfiguracja DNS

DNS Records

A Record (Address):

- Maps domain → IP address
- Przykład: `myapp. com → 20.123.45.67`
- Use case: główna domena

CNAME (Canonical Name):

- Maps subdomain → domain (alias)
- Przykład: `www. myapp.com → myapp. com`
- Use case: subdomeny, CDN

TXT (Text):

- Verification records
- Przykład: Google verification, SPF, DKIM
- Use case: domain ownership proof

MX (Mail Exchange):

- Mail server configuration
- Priority levels
- Use case: custom email (@myapp.com)

7.2 Darmowe Domeny

Free Domain Options

Freenom (. tk, .ml, .ga, .cf, .gq):

- Darmowe domeny (12 months, renewable)
- **Uwaga:** mogą być odebrane (ToS violations, inactivity)
- Nie recommended dla production (reputacja, stability)
- Use case: testing, personal projects

eu. org:

- Darmowa subdomena (yourname.eu. org)
- Stabilna, respectable
- Manual approval (może zająć dni)
- Use case: free reliable domain

DuckDNS / No-IP:

- Dynamic DNS (DDNS)
- Subdomena wskazująca na dynamic IP
- Auto-update IP (client software)
- Use case: home servers z non-static IP

GitHub Pages (username.github.io):

- Free subdomain dla static sites
- Automatic HTTPS
- Custom domain support (CNAME)

7.3 SSL Certificates

SSL/TLS Options

Let's Encrypt:

- FREE, automated, open CA
- 90-day validity (auto-renewal)
- Certbot - official client (Linux)
- win-acme - Windows client
- Use case: self-hosted servers

Cloudflare:

- FREE automatic SSL
- No installation needed (proxy mode)
- Universal SSL (wildcard)
- Auto-renewal
- Use case: easiest option

Azure:

- Free managed certificate (App Service)
- Auto-renewal
- Requires custom domain

Commercial (DigiCert, Sectigo):

- Paid certificates
- Extended Validation (EV) - green bar
- Warranty
- Use case: e-commerce, banks

8 GitHub Pages

8.1 Podstawy

GitHub Pages - Static Hosting

Czym jest:

- FREE hosting dla static websites
- Served directly z GitHub repository
- Automatic HTTPS
- Custom domain support
- Jekyll support (static site generator)

Limity:

- Static content only (HTML/CSS/JS)
- No server-side code (no . NET backend)
- Max 1 GB repo size
- 100 GB bandwidth/month (soft limit)

Use case:

- Portfolio, landing pages
- Documentation (MkDocs, Docusaurus)
- SPA (React, Vue, Angular builds)
- Blogs (Jekyll, Hugo)

8.2 Setup

Publishing na GitHub Pages

Metoda 1: username.github.io repository

- Create repo: `username.github. io`
- Push HTML files to `main` branch
- Automatic publish
- URL: `https://username.github.io`

Metoda 2: Project pages

- Any repo name
- Settings → Pages → Source: `main` branch lub `/docs` folder
- URL: `https://username.github.io/repo-name`

Custom domain:

- Add CNAME file w repo root: `myapp.com`
- DNS: A records → GitHub IPs (4 records)
- Or CNAME: `www.myapp.com → username.github.io`
- GitHub Settings → Pages → Custom domain
- Automatic SSL (Let's Encrypt)

8.3 GitHub Actions Deployment

Automatyczny Deploy

Workflow dla SPA (React/Vue):

- Build locally lub w Actions
- Output folder: `build/` lub `dist/`
- Actions: checkout → build → deploy to `gh-pages` branch
- Trigger: push to `main`

Przykład workflow:

- `. github/workflows/deploy. yml`
- Steps: `npm install → npm run build → peaceiris/actions-gh-pages`
- Automatic deployment on commit

8.4 Alternatywy

Inne Static Hosting

Netlify:

- Drag-and-drop deploy
- Automatic builds z Git
- Serverless functions (limited)
- Free tier: 100 GB bandwidth
- Custom domains, automatic SSL

Vercel:

- Optimized dla Next.js (ale też inne)
- Edge functions
- Automatic preview deployments (PR)
- Free tier: 100 GB bandwidth

Cloudflare Pages:

- Edge deployment (ultra-fast)
- Unlimited bandwidth (free)
- Workers (serverless functions)
- Git integration

9 Pytania kontrolne

1. Czym różni się hosting lokalny od VPS od PaaS?
2. Co to jest reverse proxy i jakie ma funkcje?
3. Dlaczego Kestrel powinien być za Nginx/IIS w production?
4. Co to jest port forwarding i kiedy jest potrzebny?
5. Czym jest Proxmox VE i do czego służy?
6. Wymień 3 networking modes w Proxmox i ich różnice.
7. Jaki problem rozwiązuje Cloudflare Tunnel?
8. Jak działa Cloudflare Tunnel (opisz flow)?
9. Wymień 2 darmowe opcje VPS i ich limity.
10. Co to jest Nginx i jaką pełni rolę?
11. Czym różni się Free tier od Basic tier w Azure App Service?
12. Co to jest Application Insights?
13. Czym różni się A record od CNAME?

14. Wymień 3 źródła darmowych domen.
15. Czym różni się Let's Encrypt od Cloudflare SSL?
16. Do czego służy GitHub Pages i jakie ma limity?
17. Wymień 3 alternatywy dla GitHub Pages.