# **DevOps Strategy**

## **1. Component Strategy**

### **Component: Go Proxy (intercept.prism)**

**Corresponding Source Code Repo:** prism/intercept.prism

**Deployment Location:**

* **Local Environment:** Host network (Port 7000) via Docker Compose
  + Provides direct access for local development and debugging
  + Simplifies integration with local observability tools
* **Cloud (MVP):** Microsoft Azure Container Instances (ACI)
  + Serverless container platform for simplified deployment
  + Pay-per-second billing model reduces costs during MVP phase
  + Auto-scaling capabilities for handling variable traffic

**Tests/Checks Before Deployment:**

* **Linting:** golangci-lint (Static analysis)
  + Enforces code quality standards and catches common Go pitfalls
  + Configurable ruleset for team-specific conventions
* **Automated Testing:** Integration tests in an ephemeral Docker environment
  + Ephemeral environments guarantee test isolation and reproducibility

**Build Process:**

* Multi-stage Docker builds to minimize final image size
* Separate build and runtime stages for security and efficiency
* Optimization for Go binary compilation and dependency caching

### **Component: Next.js Frontend + Backend (soul.prism)**

**Corresponding Source Code Repo:** prism/soul.prism

**Deployment Location:**

* **Local Environment:** Docker Container via Docker Compose
  + Integrated with Go backend through Docker networking
* **Cloud:** Microsoft Azure App Service (Web App for Containers)
  + Managed platform with built-in scaling and load balancing
  + Simplified SSL/TLS certificate management

**Tests/Checks Before Deployment:**

* **Formatting/Linting:** biome (Fast formatting and linting)
  + Modern, performant alternative to ESLint/Prettier
  + Unified toolchain reduces configuration complexity
  + Fast execution speeds up CI/CD pipelines
* **Automated Testing:** Integration tests ensuring logic functionality before merging to main
  + Validates business logic and API interactions
  + Gate-keeping mechanism for main branch integrity

**Build Process:**

* Next.js optimized production builds
* Static asset optimization and code splitting
* Environment-specific configuration injection

## **2. Tools, Platforms, and Libraries**

### **Source Control & CI/CD**

* **Platform:** GitHub Actions
  + Native integration with GitHub repositories
  + Extensive marketplace of pre-built actions
  + Workflow automation for builds, tests, and deployments
* **Deployment Triggers:** "Pull on Update" configured in Azure Web Apps
  + Continuous deployment from ACR to Azure services
  + Webhook-based automation reduces manual intervention
  + Automatic rollout of new container versions

**CI/CD Workflow:**

1. Code push triggers GitHub Actions workflow
2. Parallel execution of linting and testing
3. Docker image build with semantic versioning tags
4. Push to Azure Container Registry
5. Azure Web App automatically pulls and deploys new image

### **Containerization & Orchestration**

* **Docker:** Multi-stage builds for optimized images
  + Reduced image sizes improve deployment speed
  + Security benefits from minimal runtime dependencies
  + Layer caching accelerates build times
* **Docker Compose:** Local development orchestration
  + Declarative service definitions
  + Network and volume management
  + Simplified multi-container application management
* **Make:** Automation aliases and task runner
  + Standardized command interface (e.g., make build, make test)
  + Reduces cognitive load for common operations
  + Documentation through Makefile targets

**Container Registry:**

* **Azure Container Registry (ACR)**
  + Integrated security scanning
  + Azure RBAC integration for access control
  + Cost-effective for Azure-centric deployments

### **Cloud Infrastructure (Azure)**

**Compute:**

* **Azure Container Instances (ACI)** - Go Proxy
  + Fastest container startup times in Azure
  + Ideal for lightweight services
  + No cluster management overhead
  + **Consideration:** Limited to single-instance deployments initially
* **Azure App Service** - Next.js Application
  + Managed platform with auto-scaling
  + Built-in monitoring and diagnostics
  + **Consideration:** Higher cost than ACI but includes more features

**Security/Networking:**

* **Azure Virtual Network (VNet)**
  + Network isolation for backend services
  + Private connectivity between components
  + NSG (Network Security Group) rules for traffic control
* **Azure Key Vault**
  + Centralized secret management
  + Audit logging for secret access
  + Rotation policies for credentials

**Security Best Practices:**

* Managed identities for Azure resource authentication
* Principle of least privilege for service permissions
* Secrets injected as environment variables at runtime

### **Infrastructure as Code (IaC)**

* **Tools:** Bicep
  + Azure-native + first-class Azure support
  + Version-controlled infrastructure definitions
  + Reproducible environments (dev, staging, production)
  + Declarative configuration reduces manual errors

### **Observability**

* **Prometheus:** Metrics collection and storage
  + Time-series database for application metrics
  + Alerting capabilities with Alertmanager
* **Grafana:** Visualization and dashboarding
  + Rich visualization library
  + Multi-source data integration (Prometheus, Azure Monitor)
  + Customizable dashboards for different stakeholders
  + Alerting and notification system

### **DevOps Diagram**

