WARP 2.0

Full Development Lifecycle Automation and System Management

MCP Servers • Rules Engine • 1M+ Token Context Windows

Intelligent Automation from Concept to Production

OpenClaw?

<- FEEDBACK

Agenda

WARP 2.0 Fundamentals

Context windows, MD files, MCP servers, and Rules engine

Multi-Tiered Applications

UI, API, business logic, persistence, and models

Development Automation

Docker environments, testing frameworks, CI/CD

Practical Implementation

Getting started and best practices

The Development Challenge

Traditional AI Limitations

- Small context windows (8K-32K tokens)
- Can't see full project structure
- No memory between sessions
- Generic outputs without standards
- One file at a time approach
- Inconsistent patterns across files

Manual Coding Overhead

- Same patterns repeated endlessly
- Configuration is error-prone
- Each developer does it differently
- Testing added as afterthought
- Documentation quickly outdated
- 40% of time on non-coding tasks

01

WARP 2.0 Fundamentals

Context Windows, MD Files, and Intelligent Architecture

What is WARP 2.0?

Your Description

Context Engine (1M+ tokens)

MCP Server Orchestration

Rules Engine

Complete Generation

- Describe features in natural language or MD files
- Entire codebase loaded into context at once
- MCP servers provide access to all dev tools
- Your standards applied to all generated code
- Production-ready output with tests included
- WARP sees your WHOLE project, not just one file
- True multi-tasking across all application layers

Massive Context Windows (1M+ Tokens)

Capabilities

- ~750,000 words of context
- Entire medium-sized codebase
- All configuration files included
- Full documentation accessible
- Historical patterns understood
- Cross-file relationships mapped

Key Benefits

- See ALL MD files when generating new ones
- Learn naming from existing code
- Understand your architecture patterns
- Refactor across multiple files at once
- Maintain consistency automatically
- No more context limitations

MD Files for Explicit Direction

Capabilities

- architecture.md - Overall patterns
- components.md - UI conventions
- api.md - Backend standards
- testing.md - Test requirements
- docker.md - Container configs
- Custom files for your needs

Key Benefits

- Explicit instructions > implicit guessing
- Version-controlled with your code
- Team-shared standards
- WARP follows YOUR patterns
- Examples included for clarity
- Override defaults precisely

MCP Server Architecture

Capabilities

- File System - Project access
- Database - Schema management
- Docker - Container lifecycle
- Git - Version control
- Testing - Execution & reporting
- Build - Compilation & bundling

Key Benefits

- Standardized tool access protocol
- Sandboxed secure execution
- Full audit logging
- Permission controls
- Extensible with custom servers
- Orchestrated by WARP AI

The Rules Engine

Capabilities

- Code style and formatting
- Architecture pattern enforcement
- Security standards application
- Testing coverage requirements
- Documentation standards
- Naming conventions

Key Benefits

- Applied to ALL generated code
- Cascading: project > team > global
- Violations prevented before writing
- Consistent output guaranteed
- Customizable to your needs
- Version-controlled rules.yaml

02

Multi-Tiered Applications

Complete Application Generation Across All Layers

Application Architecture Overview

UI / Presentation Layer

Backend API Layer

Business Logic / Projections

Persistence Layer

Data Models / Schemas

- All layers generated from single description
- Types flow from models through to UI
- Consistent validation at every layer
- Tests generated for each layer
- Docker configured for all services
- Cross-layer refactoring supported
- True end-to-end application generation

UI / Presentation Layer

Components Generated

- Page components with routing
- Reusable UI components
- Form handling with validation
- State management setup
- API integration hooks
- Error boundaries included

MD File Controls

- Component structure patterns
- Styling approach (Tailwind, etc.)
- State management choice
- File organization rules
- Accessibility requirements
- Testing standards

Backend API Layer

Endpoints Generated

- RESTful API routes
- GraphQL schemas + resolvers
- Authentication middleware
- Authorization rules
- Input validation layers
- Rate limiting configured

Automatic Features

- OpenAPI/Swagger documentation
- Type-safe request/response
- Standardized error handling
- Logging and monitoring hooks
- Integration tests included
- Security best practices

Business Logic / Projections

Services Generated

- Service classes with DI
- Business rule validators
- Projection/transform logic
- Event handlers
- Workflow orchestration
- Computed values

Architecture Benefits

- Clean separation of concerns
- Testable with mocks
- Reusable across endpoints
- Event-sourcing support
- CQRS patterns available
- Transaction management

Persistence Layer

Repository Pattern

- Clean repository interfaces
- Query builders generated
- Transaction management
- Connection pooling setup
- Caching integration
- Migration support

ORM Support

- Prisma (primary)
- TypeORM, Sequelize
- SQLAlchemy (Python)
- Redis caching layer
- Query optimization hints
- Relationship handling

Data Models & Schemas

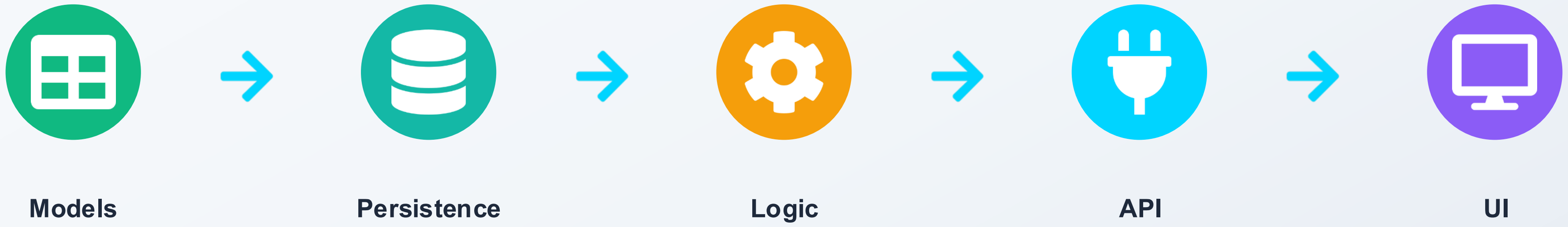
Model Generation

- Database schema from description
- Migration files (up/down)
- TypeScript/Python types
- Validation schemas (Zod)
- Relationship mappings
- Indexes for queries

Schema Evolution

- Change models, get migrations
- Type changes propagate up
- Validation updates everywhere
- Backward compatible options
- Seed data generation
- Full audit trail

Cross-Layer Integration



Types flow from models through every layer - change once, updates everywhere. Validation rules defined at schema level apply consistently through API and UI. Error handling standardized across all layers. Because WARP sees your entire project in its 1M+ token context, it maintains consistency that manual coding simply cannot achieve. This is the power of true multi-tasking.

03

Development Automation

Docker, Testing, and CI/CD Integration

Docker & Environment Automation

Capabilities

- Optimized Dockerfiles
- docker-compose for local dev
- Multi-stage builds
- Health checks configured
- Environment variable management
- Kubernetes manifests ready

Key Benefits

- DEV: Hot reload, debug mode
- TEST: Isolated, seeded data
- STAGING: Production-like
- PROD: Optimized, secured
- MD files control all configs
- One-command environment setup

Testing Framework Automation

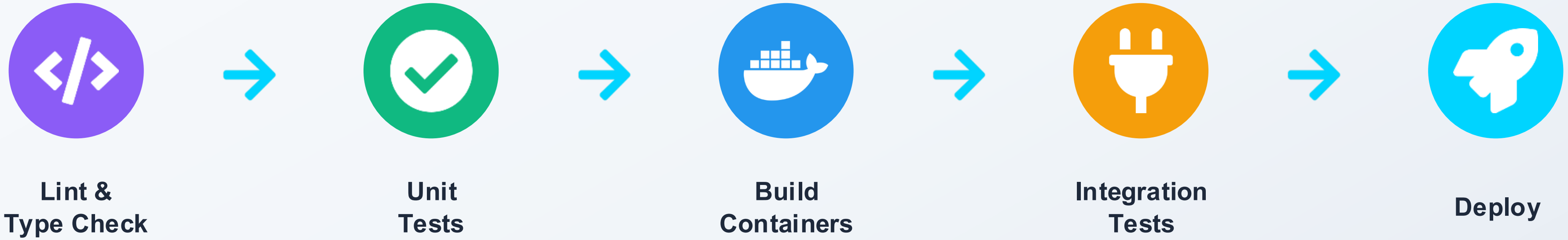
Capabilities

- Unit tests (Jest/Vitest)
- API tests (Supertest)
- E2E tests (Cypress/Playwright)
- Test fixtures and factories
- Coverage reporting
- Snapshot testing

Key Benefits

- Tests generated WITH code
- Coverage requirements enforced
- Testing pyramid followed
- Mocks auto-configured
- CI/CD integration included
- Visual regression optional

CI/CD Pipeline Generation



Complete pipeline generation for GitHub Actions, GitLab CI, Jenkins, or CircleCI. Includes lint/type checking, unit tests, container builds, integration tests, staging deployment, E2E tests, and production deployment with approval gates. Automatic rollback on failed health checks. Environment promotion handled automatically.

04

Practical Implementation

Getting Started with WARP 2.0

Getting Started

Step 1: Setup

Install WARP CLI, initialize project, configure MCP servers

Step 2: Create MD Files

Define architecture.md, components.md, api.md, testing.md

Step 3: Describe Your App

Natural language description of features and behavior

Step 4: Review & Iterate

Review generated code, refine MD files, regenerate as needed

Best Practices

MD File Tips

Be specific about patterns, include examples, version control with code

Effective Prompting

Describe behavior not implementation, include constraints, reference patterns

Incremental Approach

Generate layer by layer initially, build confidence, then full apps

Review Process

WARP is a tool not replacement, review architecture, customize logic

Thank You

Questions & Live Demo

1M+ Context • MD Files • Multi-Tier Generation

Streamline your entire development process
from initial concept through production.

<- FEEDBACK