HelixCode

A distributed, AI-powered software development platform with multi-platform support.

Features

- **Multi-Platform Support**: Desktop, mobile, terminal, and specialized OS clients
- **Distributed Computing**: Worker nodes for parallel task execution
- **AI Integration**: LLM-powered code generation and reasoning
- **Real-time Collaboration**: MCP protocol for tool execution
- **Authentication & Security**: JWT-based auth with session management
- **Task Management**: Checkpoint-based work preservation
- **Notification System**: Multi-channel notifications (Slack, Email, Discord)

Quick Start

Development

```
# Clone the repository
git clone https://github.com/your-org/helixcode.git
cd helixcode

# Install dependencies
go mod download

# Generate assets
make logo-assets

# Build the server
make build

# Run tests
make test

# Start development server
make dev
```

Production Deployment

1. **Clone and setup:**

```
git clone https://github.com/your-org/helixcode.git
cd helixcode
cp .env.example .env
```

2. **Configure environment variables:** Edit `.env` file with your production values:

```
HELIX_AUTH_JWT_SECRET=your-super-secure-jwt-secret
HELIX_DATABASE_PASSWORD=your-secure-database-password
HELIX_REDIS_PASSWORD=your-secure-redis-password
```

3. **Deploy with Docker Compose:**

```
docker-compose up -d
```

4. **Check deployment:**

```
docker-compose ps
curl http://localhost/health
```

Architecture

Core Components

- **Server**: Main API server with REST and WebSocket endpoints
- **Database**: PostgreSQL for persistent data storage
- **Cache**: Redis for session and task state management
- **Workers**: Distributed worker nodes for task execution
- **MCP Server**: Model Context Protocol for AI tool integration

Applications

- **Desktop**: Full-featured desktop application (Fyne)
- **Terminal UI**: Terminal-based interface (tview)
- **Aurora OS**: Specialized Aurora OS client
- **Symphony OS**: Optimized Symphony OS client
- **Mobile**: Cross-platform mobile applications

Configuration

Configuration is managed through YAML files and environment variables. See `config/config.yaml` for default settings.

Key configuration areas:

- Server settings (ports, timeouts)
- Database connection
- Redis configuration
- Authentication settings
- Worker management
- LLM provider settings

API Documentation

Authentication Endpoints

- `POST /api/auth/register` - User registration
- `POST /api/auth/login` - User login
- `POST /api/auth/logout` - User logout
- `POST /api/auth/refresh` - Token refresh
- `GET /api/auth/me` - Current user info

Task Management

- `GET /api/tasks` - List tasks
- `POST /api/tasks` - Create task
- `GET /api/tasks/{id}` - Get task details
- `PUT /api/tasks/{id}` - Update task
- `DELETE /api/tasks/{id}` - Delete task

Worker Management

- `GET /api/workers` - List workers
- `POST /api/workers` - Register worker
- `GET /api/workers/{id}` - Get worker details
- `DELETE /api/workers/{id}` - Remove worker

Development

Building Applications

```
# Build all applications
make prod

# Build specific applications
make aurora-os
make symphony-os

# Build mobile bindings
make mobile-ios
make mobile-android
```

Testing

```
# Run all tests
make test

# Run specific test suites
go test ./internal/auth/...
go test ./internal/worker/...

# Run with coverage
go test -cover ./...
```

Code Quality

```
# Format code
make fmt

# Lint code
make lint
```

Deployment Options

Docker Compose (Recommended)

The production `docker-compose.yml` includes:

- HelixCode server
- PostgreSQL database
- Redis cache
- Nginx reverse proxy
- Prometheus monitoring
- Grafana dashboards

Manual Deployment

1. Build the binary: `make prod`
2. Setup PostgreSQL and Redis
3. Configure environment variables
4. Run the server: `./bin/helixcode-server`

Kubernetes

For large-scale deployments, use the provided Kubernetes manifests in the `k8s/` directory.

Monitoring

The deployment includes Prometheus and Grafana for monitoring:

- Application metrics

- Database performance
- Worker health
- Task execution stats

Access Grafana at `http://localhost:3000` (default credentials: admin/admin)

Security

- JWT-based authentication
- Password hashing with bcrypt
- SSH key-based worker authentication
- Environment variable configuration
- No secrets in code or config files

Contributing

1. Fork the repository
2. Create a feature branch
3. Make your changes
4. Add tests
5. Submit a pull request

License

This project is licensed under the terms specified in the LICENSE file.

Support

For support and questions:

- GitHub Issues: https://github.com/your-org/helixcode/issues
- Documentation: https://docs.helixcode.dev
- Community: https://community.helixcode.dev