FLY.io Deployment & Chrome Extension Integration Guide

Part 1: FLY.io Deployment

Prerequisites

- FLY.io account (create at https://fly.io)
- FLY CLI installed (`curl -L https://fly.io/install.sh | sh`)

Step 1: Create Dockerfile

Create `Dockerfile` in your server directory:

dockerfile

```dockerfile
# Dockerfile
FROM node:20-alpine

# Install SQLite
RUN apk add --no-cache sqlite

WORKDIR /app

# Copy package files
COPY package*.json ./
RUN npm ci --only=production

# Copy application files
COPY . .

# Create necessary directories
RUN mkdir -p data uploads logs

# Copy database if exists
COPY data/frameworks.db data/frameworks.db

# Expose port
EXPOSE 3000

# Start the application
CMD ["node", "server.js"]
```

Step 2: Create fly.toml Configuration

Create `fly.toml` in your server directory:

toml

```toml
# fly.toml
app = "promptinstyl"
primary_region = "ord"

[build]
  dockerfile = "Dockerfile"

[env]
  NODE_ENV = "production"
  PORT = "3000"

[experimental]
  auto_rollback = true

[[services]]
  http_checks = []
  internal_port = 3000
  processes = ["app"]
  protocol = "tcp"
  script_checks = []

  [services.concurrency]
    hard_limit = 25
    soft_limit = 20
    type = "connections"

  [[services.ports]]
    force_https = true
    handlers = ["http"]
    port = 80

  [[services.ports]]
    handlers = ["tls", "http"]
    port = 443

  [[services.tcp_checks]]
    grace_period = "1s"
    interval = "15s"
    restart_limit = 0
    timeout = "2s"

[mounts]
```

```
    source="promptinstyl_data"
    destination="/app/data"
```

Step 3: Deploy to FLY.io

```bash
# Login to FLY
fly auth login

# Launch app (first time)
fly launch

# Set secrets
fly secrets set OPENAI_API_KEY=sk-your-key-here

# Create volume for persistent data
fly volumes create promptinstyl_data --size 1

# Deploy
fly deploy

# Check status
fly status

# View logs
fly logs
```

Step 4: Update server.js for Production

Add to your `server.js`:

```javascript
// Health check for FLY.io
app.get('/_health', (req, res) => {
  res.status(200).send('OK');
});

// Trust proxy for proper IP handling
app.set('trust proxy', true);
```

Part 2: Chrome Extension Development

Extension Structure

Create a new directory `chrome-extension/` with:

```
chrome-extension/
├── manifest.json
├── popup.html
├── popup.js
├── popup.css
├── content.js
├── background.js
└── icons/
    ├── icon16.png
    ├── icon48.png
    └── icon128.png
```

manifest.json

json

```
{
  "manifest_version": 3,
  "name": "PromptInSTYL",
  "version": "1.0.0",
  "description": "AI-powered prompt engineering assistant",

  "permissions": [
    "activeTab",
    "storage",
    "contextMenus"
  ],

  "host_permissions": [
    "https://promptinstyl.fly.dev/*",
    "http://localhost:3000/*"
  ],

  "action": {
    "default_popup": "popup.html",
    "default_icon": {
      "16": "icons/icon16.png",
      "48": "icons/icon48.png",
      "128": "icons/icon128.png"
    }
  },

  "background": {
    "service_worker": "background.js"
  },

  "content_scripts": [
    {
      "matches": ["<all_urls>"],
      "js": ["content.js"],
      "run_at": "document_idle"
    }
  ],

  "icons": {
    "16": "icons/icon16.png",
    "48": "icons/icon48.png",
    "128": "icons/icon128.png"
```

```
  }
}
```

popup.html

html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <link rel="stylesheet" href="popup.css">
</head>
<body>
  <div class="container">
    <h1>PromptInSTYL</h1>

    <div class="input-section">
      <textarea id="userRequest" placeholder="What do you need help with?" rows="4"></textarea>
      <button id="analyzeBtn">Analyze Intent</button>
    </div>

    <div id="loading" class="loading hidden">Analyzing...</div>

    <div id="results" class="results hidden">
      <h3>Analysis Results</h3>
      <div id="intent"></div>
      <div id="framework"></div>
      <div id="confidence"></div>
      <button id="copyPromptBtn" class="hidden">Copy Prompt</button>
    </div>

    <div class="settings">
      <label>
        <input type="checkbox" id="useSelection"> Use selected text
      </label>
      <a href="#" id="settingsLink">Settings</a>
    </div>
  </div>

  <script src="popup.js"></script>
</body>
</html>
```

popup.css

css

```css
body {
  width: 400px;
  min-height: 300px;
  margin: 0;
  padding: 0;
  font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, sans-serif;
}

.container {
  padding: 16px;
}

h1 {
  font-size: 20px;
  margin: 0 0 16px 0;
  color: #1a1a1a;
}

.input-section {
  margin-bottom: 16px;
}

textarea {
  width: 100%;
  padding: 8px;
  border: 1px solid #ddd;
  border-radius: 4px;
  font-size: 14px;
  resize: vertical;
  box-sizing: border-box;
}

button {
  width: 100%;
  padding: 10px;
  background: #4CAF50;
  color: white;
  border: none;
  border-radius: 4px;
  font-size: 16px;
  cursor: pointer;
  margin-top: 8px;
}
```

```css
button:hover {
  background: #45a049;
}

button:disabled {
  background: #ccc;
  cursor: not-allowed;
}

.loading {
  text-align: center;
  color: #666;
  padding: 20px;
}

.hidden {
  display: none;
}

.results {
  background: #f5f5f5;
  padding: 12px;
  border-radius: 4px;
  margin-bottom: 16px;
}

.results h3 {
  margin: 0 0 12px 0;
  font-size: 16px;
}

.results div {
  margin-bottom: 8px;
  font-size: 14px;
}

.settings {
  display: flex;
  justify-content: space-between;
  align-items: center;
  font-size: 14px;
}
```

```css
.settings label {
  display: flex;
  align-items: center;
  gap: 6px;
}

.error {
  color: #d32f2f;
  padding: 12px;
  background: #ffebee;
  border-radius: 4px;
  margin-bottom: 16px;
  font-size: 14px;
}
```

popup.js

javascript

```
// popup.js
const API_URL = 'https://promptinstyl.fly.dev/api'; // Update with your FLY.io URL
// const API_URL = 'http://localhost:3000/api'; // For development

document.addEventListener('DOMContentLoaded', async () => {
  const userRequestEl = document.getElementById('userRequest');
  const analyzeBtnEl = document.getElementById('analyzeBtn');
  const loadingEl = document.getElementById('loading');
  const resultsEl = document.getElementById('results');
  const useSelectionEl = document.getElementById('useSelection');

  // Check if there's selected text
  const [tab] = await chrome.tabs.query({ active: true, currentWindow: true });
  chrome.tabs.sendMessage(tab.id, { action: 'getSelection' }, (response) => {
    if (response && response.text) {
      userRequestEl.value = response.text;
      useSelectionEl.checked = true;
    }
  });

  // Load saved settings
  chrome.storage.sync.get(['apiUrl'], (data) => {
    if (data.apiUrl) {
      API_URL = data.apiUrl;
    }
  });

  analyzeBtnEl.addEventListener('click', async () => {
    const userRequest = userRequestEl.value.trim();

    if (userRequest.length < 5) {
      showError('Please enter at least 5 characters');
      return;
    }

    analyzeBtnEl.disabled = true;
    loadingEl.classList.remove('hidden');
    resultsEl.classList.add('hidden');

    try {
      const response = await fetch(`${API_URL}/analyze-intent`, {
        method: 'POST',
        headers: {
```

```
            'Content-Type': 'application/json'
          },
          body: JSON.stringify({ userRequest })
        });

        if (!response.ok) {
          throw new Error(`HTTP error! status: ${response.status}`);
        }

        const data = await response.json();

        if (data.success) {
          displayResults(data.data);
        } else {
          showError(data.error || 'Analysis failed');
        }

      } catch (error) {
        showError(`Error: ${error.message}`);
      } finally {
        analyzeBtnEl.disabled = false;
        loadingEl.classList.add('hidden');
      }
    });

    function displayResults(data) {
      document.getElementById('intent').innerHTML =
        `<strong>Intent:</strong> ${data.analysis.intent}`;

      document.getElementById('framework').innerHTML =
        `<strong>Framework:</strong> ${data.framework.selected.name}`;

      document.getElementById('confidence').innerHTML =
        `<strong>Confidence:</strong> ${data.metadata.confidence.overall}%`;

      resultsEl.classList.remove('hidden');

      // Store the full prompt for copying
      if (data.prompt && data.prompt.prompt) {
        const copyBtn = document.getElementById('copyPromptBtn');
        copyBtn.classList.remove('hidden');
        copyBtn.onclick = () => {
          navigator.clipboard.writeText(data.prompt.prompt);
          copyBtn.textContent = 'Copied!';
```

```
        setTimeout(() => {
          copyBtn.textContent = 'Copy Prompt';
        }, 2000);
      };
    }
  }

  function showError(message) {
    const errorEl = document.createElement('div');
    errorEl.className = 'error';
    errorEl.textContent = message;
    resultsEl.parentNode.insertBefore(errorEl, resultsEl);

    setTimeout(() => {
      errorEl.remove();
    }, 5000);
  }

  // Settings link
  document.getElementById('settingsLink').addEventListener('click', (e) => {
    e.preventDefault();
    chrome.runtime.openOptionsPage();
  });
});
```

content.js

javascript

```javascript
// content.js
chrome.runtime.onMessage.addListener((request, sender, sendResponse) => {
  if (request.action === 'getSelection') {
    const selectedText = window.getSelection().toString();
    sendResponse({ text: selectedText });
  }
  return true;
});

// Context menu integration
document.addEventListener('mouseup', (e) => {
  const selectedText = window.getSelection().toString();
  if (selectedText.length > 5) {
    chrome.runtime.sendMessage({
      action: 'updateContextMenu',
      selectedText: selectedText
    });
  }
});
```

background.js

javascript

```javascript
// background.js
chrome.runtime.onInstalled.addListener(() => {
  // Create context menu
  chrome.contextMenus.create({
    id: 'promptinstyl',
    title: 'Analyze with PromptInSTYL',
    contexts: ['selection']
  });
});

// Handle context menu clicks
chrome.contextMenus.onClicked.addListener((info, tab) => {
  if (info.menuItemId === 'promptinstyl' && info.selectionText) {
    // Store the selected text
    chrome.storage.local.set({
      pendingText: info.selectionText
    }, () => {
      // Open popup
      chrome.action.openPopup();
    });
  }
});

// Handle messages from content script
chrome.runtime.onMessage.addListener((request, sender, sendResponse) => {
  if (request.action === 'updateContextMenu' && request.selectedText) {
    chrome.contextMenus.update('promptinstyl', {
      title: `Analyze "${request.selectedText.substring(0, 30)}..." with PromptInSTYL`
    });
  }
});
```

Part 3: Testing & Deployment

Testing Chrome Extension Locally

1. Open Chrome and go to `chrome://extensions/`
2. Enable "Developer mode"
3. Click "Load unpacked"
4. Select your `chrome-extension` directory

5. The extension icon should appear in your toolbar

Publishing to Chrome Web Store

1. Create a developer account ($5 one-time fee)
2. Zip your extension files
3. Upload to Chrome Web Store Dashboard
4. Fill in listing details
5. Submit for review

Connecting Extension to FLY.io Backend

Update `popup.js` to use your FLY.io URL:

```javascript
const API_URL = 'https://promptinstyl.fly.dev/api';
```

CORS Configuration

Update your `server.js` to allow extension requests:

```javascript
app.use(cors({
  origin: [
    'chrome-extension://your-extension-id',
    'https://promptinstyl.fly.dev',
    'http://localhost:3000'
  ],
  credentials: true
}));
```

Part 4: Advanced Features

1. Auto-inject into AI Chat Interfaces

Add to `content.js`:

javascript

```javascript
// Detect ChatGPT, Claude, etc.
const chatDetectors = {
  chatgpt: () => document.querySelector('textarea[data-id]'),
  claude: () => document.querySelector('div[contenteditable="true"]'),
  bard: () => document.querySelector('rich-textarea')
};

// Inject analyzed prompt
function injectPrompt(prompt) {
  for (const [platform, detector] of Object.entries(chatDetectors)) {
    const element = detector();
    if (element) {
      if (element.tagName === 'TEXTAREA') {
        element.value = prompt;
        element.dispatchEvent(new Event('input', { bubbles: true }));
      } else {
        element.textContent = prompt;
        element.dispatchEvent(new Event('input', { bubbles: true }));
      }
      break;
    }
  }
}
```

2. Keyboard Shortcuts

Add to `manifest.json`:

```json
"commands": {
  "_execute_action": {
    "suggested_key": {
      "default": "Ctrl+Shift+P",
      "mac": "Command+Shift+P"
    }
  },
  "analyze-selection": {
    "suggested_key": {
      "default": "Ctrl+Shift+A"
    },
    "description": "Analyze selected text"
  }
}
```

3. Options Page

Create `options.html`:

html

```html
<!DOCTYPE html>
<html>
<head>
  <title>PromptInSTYL Settings</title>
  <style>
    body { padding: 20px; font-family: Arial, sans-serif; }
    .setting { margin-bottom: 15px; }
    input[type="text"] { width: 300px; padding: 5px; }
    button { padding: 8px 15px; background: #4CAF50; color: white; border: none; cursor: pointe
  </style>
</head>
<body>
  <h1>PromptInSTYL Settings</h1>

  <div class="setting">
    <label>API URL:</label><br>
    <input type="text" id="apiUrl" placeholder="https://promptinstyl.fly.dev/api">
  </div>

  <div class="setting">
    <label>
      <input type="checkbox" id="autoAnalyze">
      Automatically analyze selected text
    </label>
  </div>

  <button id="save">Save Settings</button>
  <div id="status"></div>

  <script src="options.js"></script>
</body>
</html>
```

Deployment Checklist

FLY.io

- [ ] Create Dockerfile
- [ ] Configure fly.toml
- [ ] Set environment variables
- [ ] Create persistent volume

- [ ] Deploy and test
- [ ] Set up custom domain (optional)

Chrome Extension

- [ ] Create all required files
- [ ] Add icons (16x16, 48x48, 128x128)
- [ ] Test locally
- [ ] Update API URL to production
- [ ] Submit to Chrome Web Store

Total Time Estimate

- FLY.io deployment: 30 minutes
- Chrome extension basic: 1 hour
- Chrome extension advanced: 2-3 hours
- Testing & refinement: 1 hour

Total: 4-5 hours for full implementation