GitHub Actions AI-CI Templates Collection
Copy these files to your .github/workflows/ directory

 ==
Template 1: ai-merge-predictor.yml
Predicts and prevents merge conflicts
==

```
name: AI Merge Conflict Prediction

on:
  pull_request:
    types: [opened, synchronize, reopened]

jobs:
  conflict-prediction:
    runs-on: ubuntu-latest
    timeout-minutes: 10

    steps:
      - name: Checkout code
        uses: actions/checkout@v4
        with:
          fetch-depth: 0
          token: ${{ secrets.GITHUB_TOKEN }}

      - name: Setup Node.js
        uses: actions/setup-node@v4
        with:
          node-version: '18'
          cache: 'npm'

      - name: Install AI Conflict Predictor
        run: |
          # Replace with actual tool when available
          npm install -g merge-conflict-ai
          echo "🤖 AI conflict predictor installed"

      - name: Get Changed Files
        id: changed-files
        uses: tj-actions/changed-files@v40
        with:
          files: |
            src/**
            lib/**
            config/**
          files_ignore: |
            docs/**
            *.md
            *.txt

      - name: Analyze Merge Conflicts
        id: conflict-analysis
        run: |
          echo "🔍 Analyzing potential conflicts..."

          # Create mock analysis (replace with real AI tool)
          cat > conflict-analysis.json << EOF
          {
            "riskLevel": "medium",
            "riskScore": 0.6,
            "predictedConflicts": [
              {
                "file": "src/components/UserForm.js",
                "type": "variable_naming",
                "confidence": 0.8,
                "suggestion": "Consider renaming 'userData' to 'userFormData' to avoid conflicts"
              }
            ],
            "recommendations": [
              "Review variable naming in UserForm.js",
              "Check database migration order",
              "Verify API endpoint changes don't break existing calls"
            ],
            "autoFixAvailable": true
          }
          EOF

          # Set outputs for next steps
          risk_score=$(cat conflict-analysis.json | jq -r '.riskScore')
          echo "risk-score=$risk_score" >> $GITHUB_OUTPUT

      - name: Auto-Fix Low Risk Issues
        if: steps.conflict-analysis.outputs.risk-score < '0.3'
        run: |
          echo "✅ Auto-fixing low-risk conflicts..."
          # Add auto-fix logic here

      - name: Create PR Comment
        uses: actions/github-script@v7
        with:
          script: |
            const fs = require('fs');
            const analysis = JSON.parse(fs.readFileSync('conflict-analysis.json', 'utf8'));

            const riskEmoji = analysis.riskScore < 0.3 ? '🟢' : analysis.riskScore < 0.7 ? '🟡' : '🔴';

            const comment = `## ${riskEmoji} AI Merge Conflict Analysis

            **Risk Level**: ${analysis.riskLevel.toUpperCase()}
            **Risk Score**: ${(analysis.riskScore * 100).toFixed(0)}%
            **Predicted Conflicts**: ${analysis.predictedConflicts.length}

            ### 🔍 Analysis Results
            ${analysis.predictedConflicts.map(conflict =>
              `- **${conflict.file}**: ${conflict.suggestion} (${(conflict.confidence * 100).toFixed(0)}%
confidence)`
            ).join('\n')}

            ### 💡 Recommendations
            ${analysis.recommendations.map(rec => `- ${rec}`).join('\n')}

            ${analysis.autoFixAvailable ? '✅ Auto-fix suggestions will be applied automatically for
low-risk items.' : '⚠️ Manual review required before merge.'}

            ---
            *Generated by AI Merge Predictor - Radiant Agility*`;

            github.rest.issues.createComment({
              issue_number: context.issue.number,
              owner: context.repo.owner,
              repo: context.repo.repo,
              body: comment
            });
      - name: Slack Alert for High Risk
        if: steps.conflict-analysis.outputs.risk-score > '0.7'
        uses: rtCamp/action-slack-notify@v2
        env:
          SLACK_WEBHOOK: ${{ secrets.SLACK_WEBHOOK }}
          SLACK_COLOR: warning
          SLACK_MESSAGE: |
            🚨 High-risk merge conflict detected in PR #${{ github.event.pull_request.number }}
            Repository: ${{ github.repository }}
            Author: ${{ github.event.pull_request.user.login }}
            Review required before merge.
```

==

Template 2: ai-test-coverage.yml
 Intelligent test coverage analysis
==

```
name: AI Test Coverage Intelligence

on:
  pull_request:
    types: [opened, synchronize]
  push:
    branches: [main, develop]

jobs:
  test-intelligence:
    runs-on: ubuntu-latest
    timeout-minutes: 15

    steps:
      - name: Checkout code
        uses: actions/checkout@v4

      - name: Setup Node.js
        uses: actions/setup-node@v4
        with:
          node-version: '18'
          cache: 'npm'

      - name: Install dependencies
```

```
        run: npm ci

      - name: Get Changed Files
        id: changed-files
        uses: tj-actions/changed-files@v40
        with:
          files: |
            src/**/*.{js,ts,jsx,tsx}
            lib/**/*.{js,ts,jsx,tsx}

      - name: Run Existing Tests
        run: |
          npm run test:coverage
          echo "📊 Current test coverage calculated"

      - name: AI Test Gap Analysis
        id: test-analysis
        run: |
          echo "🤖 Analyzing test coverage gaps..."

          # Create mock test analysis (replace with real AI tool)
          cat > test-analysis.json << EOF
          {
            "overallCoverage": 78,
            "changedFilesCoverage": 65,
            "criticalGaps": [
              {
                "file": "src/utils/payment.js",
                "functions": ["validateCreditCard", "processPayment"],
                "riskLevel": "high",
                "businessImpact": "Payment processing failures could cause revenue loss"
              }
            ],
            "suggestedTests": [
              {
                "file": "src/utils/payment.js",
                "testType": "unit",
                "description": "Test credit card validation with invalid inputs",
                "priority": "high",
                "generatedTest": "// Auto-generated test\ntest('should reject invalid credit card
numbers', () => {\n  expect(validateCreditCard('1234')).toBe(false);\n});"
              }
            ],
            "coverageTarget": 85,
            "estimatedEffort": "15 minutes"
          }
          EOF

          coverage=$(cat test-analysis.json | jq -r '.changedFilesCoverage')
          echo "coverage=$coverage" >> $GITHUB_OUTPUT

      - name: Generate Missing Tests
        if: steps.test-analysis.outputs.coverage < '80'
        run: |
          echo "🔧 Generating missing tests..."
          mkdir -p tests/auto-generated

          # Extract suggested tests and create files
          cat test-analysis.json | jq -r '.suggestedTests[].generatedTest' >
tests/auto-generated/payment.test.js
          echo "✅ Auto-generated tests created"

      - name: Run Generated Tests
        if: steps.test-analysis.outputs.coverage < '80'
        run: |
          echo "🧪 Running generated tests..."
          npm test tests/auto-generated/
          echo "✅ Generated tests passing"

      - name: Create Test Coverage Report
        uses: actions/github-script@v7
        with:
          script: |
            const fs = require('fs');
            const analysis = JSON.parse(fs.readFileSync('test-analysis.json', 'utf8'));

            const coverageEmoji = analysis.changedFilesCoverage >= 80 ? '🟢' :
analysis.changedFilesCoverage >= 60 ? '🟡' : '🔴';

            const comment = `## ${coverageEmoji} AI Test Coverage Analysis

            **Overall Coverage**: ${analysis.overallCoverage}%
            **Changed Files Coverage**: ${analysis.changedFilesCoverage}%
            **Target Coverage**: ${analysis.coverageTarget}%

            ### 🚨 Critical Coverage Gaps
            ${analysis.criticalGaps.map(gap =>
             `- **${gap.file}**: Missing tests for \`${gap.functions.join(', ')}\`\n  - Risk:
${gap.riskLevel}\n  - Impact: ${gap.businessImpact}`
            ).join('\n\n')}

            ### 🤖 AI-Generated Test Suggestions
            ${analysis.suggestedTests.map(test =>
              `- **${test.file}**: ${test.description} (${test.priority} priority)`
            ).join('\n')}

            ${analysis.changedFilesCoverage < 80 ?
              `### ✅ Auto-Generated Tests\nI've created ${analysis.suggestedTests.length} test(s)
to improve coverage. Estimated effort: ${analysis.estimatedEffort}` :
              '### ✅ Coverage Target Met\nNo additional tests required.'}

            ---
            *Generated by AI Test Intelligence - Radiant Agility*`;

            github.rest.issues.createComment({
              issue_number: context.issue.number || 'N/A',
              owner: context.repo.owner,
              repo: context.repo.repo,
              body: comment
            });
```

==
Template 3: ai-deployment-intelligence.yml
Smart deployment risk assessment
 ==

```
name: AI Deployment Intelligence

on:
  push:
    branches: [main, production, staging]
  workflow_dispatch:
    inputs:
      environment:
        description: 'Target environment'
        required: true
        default: 'staging'
        type: choice
        options:
```

```
        - staging
        - production

jobs:
  deployment-risk-assessment:
    runs-on: ubuntu-latest
    timeout-minutes: 10

    steps:
      - name: Checkout code
        uses: actions/checkout@v4
        with:
          fetch-depth: 10

      - name: Setup Node.js
        uses: actions/setup-node@v4
        with:
          node-version: '18'

      - name: Determine Environment
        id: environment
        run: |
          if [[ "${{ github.event_name }}" == "workflow_dispatch" ]]; then
            echo "env=${{ github.event.inputs.environment }}" >> $GITHUB_OUTPUT
          elif [[ "${{ github.ref }}" == "refs/heads/production" ]]; then
            echo "env=production" >> $GITHUB_OUTPUT
          elif [[ "${{ github.ref }}" == "refs/heads/main" ]]; then
            echo "env=staging" >> $GITHUB_OUTPUT
          else
            echo "env=development" >> $GITHUB_OUTPUT
          fi

      - name: Analyze Deployment Risk
        id: risk-analysis
        run: |
          echo "🤖 Analyzing deployment risk for ${{ steps.environment.outputs.env }}..."

          # Get commit information
          commit_count=$(git rev-list --count HEAD~5..HEAD)
          changed_files=$(git diff --name-only HEAD~1 HEAD | wc -l)

          # Create mock risk analysis (replace with real AI tool)
          cat > deployment-risk.json << EOF
          {
```

```
      "environment": "${{ steps.environment.outputs.env }}",
      "riskScore": 45,
      "riskLevel": "medium",
      "factors": [
        {
          "category": "code_changes",
          "impact": "medium",
          "description": "${changed_files} files changed in latest commit",
          "weight": 0.3
        },
        {
          "category": "timing",
          "impact": "low",
          "description": "Deployment during business hours",
          "weight": 0.2
        },
        {
          "category": "dependencies",
          "impact": "low",
          "description": "No dependency updates detected",
          "weight": 0.1
        }
      ],
      "recommendations": [
        "Monitor application metrics for 30 minutes post-deployment",
        "Have rollback plan ready",
        "Deploy during low-traffic hours if possible"
      ],
      "autoApproved": false,
      "requiresManualReview": true,
      "estimatedRollbackTime": "5 minutes"
    }
    EOF

    risk_score=$(cat deployment-risk.json | jq -r '.riskScore')
    echo "risk-score=$risk_score" >> $GITHUB_OUTPUT
    echo "environment=${{ steps.environment.outputs.env }}" >> $GITHUB_OUTPUT

- name: Check Business Hours
  id: business-hours
  run: |
    current_hour=$(date +%H)
    current_day=$(date +%u)  # 1=Monday, 7=Sunday
```

```
        if [[ $current_day -le 5 && $current_hour -ge 9 && $current_hour -le 17 ]]; then
          echo "in-business-hours=true" >> $GITHUB_OUTPUT
        else
          echo "in-business-hours=false" >> $GITHUB_OUTPUT
        fi

    - name: Auto-Approve Low Risk Deployment
      if: steps.risk-analysis.outputs.risk-score < '30' && steps.environment.outputs.env !=
'production'
      run: |
        echo "✅ Low risk deployment auto-approved"
        echo "DEPLOYMENT_APPROVED=true" >> $GITHUB_ENV

    - name: Require Manual Review
      if: steps.risk-analysis.outputs.risk-score >= '30' || steps.environment.outputs.env ==
'production'
      run: |
        echo "⚠️ Manual review required for deployment"
        echo "DEPLOYMENT_APPROVED=false" >> $GITHUB_ENV

    - name: Create Deployment Issue
      if: env.DEPLOYMENT_APPROVED == 'false'
      uses: actions/github-script@v7
      with:
        script: |
          const fs = require('fs');
          const analysis = JSON.parse(fs.readFileSync('deployment-risk.json', 'utf8'));

          const issue = await github.rest.issues.create({
            owner: context.repo.owner,
            repo: context.repo.repo,
            title: `🚀 Deployment Review Required: ${analysis.environment}`,
            body: `## Deployment Risk Assessment

            **Environment**: ${analysis.environment}
            **Risk Score**: ${analysis.riskScore}/100 (${analysis.riskLevel})
            **Commit**: ${context.sha.substring(0, 7)}
            **Triggered by**: ${context.actor}

            ### Risk Factors
            ${analysis.factors.map(factor =>
              `- **${factor.category}**: ${factor.impact} impact - ${factor.description}`
            ).join('\n')}
        ### 💡 Recommendations
        ${analysis.recommendations.map(rec => `- ${rec}`).join('\n')}

        ### Actions Required
        - [ ] Review code changes
        - [ ] Verify test coverage
        - [ ] Confirm rollback plan
        - [ ] Monitor post-deployment

        **Estimated Rollback Time**: ${analysis.estimatedRollbackTime}

        ---
        Comment "APPROVE" to proceed with deployment.
        Comment "REJECT" to cancel deployment.`,
        labels: ['deployment', 'review-required', analysis.riskLevel + '-risk']
      });

      console.log('Created deployment review issue:', issue.data.number);

- name: Proceed with Low Risk Deployment
  if: env.DEPLOYMENT_APPROVED == 'true'
  run: |
    echo "🚀 Proceeding with deployment..."
    # Add your deployment commands here
    # Example:
    # npm run build
    # npm run deploy:${{ steps.environment.outputs.env }}

- name: Post-Deployment Monitoring Setup
  if: env.DEPLOYMENT_APPROVED == 'true'
  run: |
    echo "📊 Setting up post-deployment monitoring..."
    # Add monitoring setup commands
    # Example: setup health checks, alerts, etc.

- name: Slack Deployment Notification
  if: always()
  uses: rtCamp/action-slack-notify@v2
  env:
    SLACK_WEBHOOK: ${{ secrets.SLACK_WEBHOOK }}
    SLACK_COLOR: ${{ env.DEPLOYMENT_APPROVED == 'true' && 'good' || 'warning' }}
    SLACK_MESSAGE: |
      🚀 Deployment to ${{ steps.environment.outputs.env }}
```

            Status: ${{ env.DEPLOYMENT_APPROVED == 'true' && 'Auto-approved' || 'Requires
review' }}
            Risk Score: ${{ steps.risk-analysis.outputs.risk-score }}/100
            Commit: ${{ github.sha }}

            ${{ env.DEPLOYMENT_APPROVED == 'false' && 'Manual review required before
proceeding.' || 'Deployment completed successfully.' }}

 ==
Template 4: ai-ci-dashboard.yml
Generates CI/CD intelligence dashboard
==

```
name: AI CI/CD Dashboard

on:
  schedule:
    - cron: '0 9 * * MON'  # Every Monday at 9 AM
  workflow_dispatch:

jobs:
  generate-dashboard:
    runs-on: ubuntu-latest

    steps:
      - name: Checkout code
        uses: actions/checkout@v4

      - name: Collect CI/CD Metrics
        id: collect-metrics
        run: |
```