JIRA Integration Setup Guide

Phase 1: GitHub for JIRA Integration

Step 1: Install GitHub for JIRA App

1. Go to Atlassian Marketplace
2. Search "GitHub for JIRA"
3. Install to your `tobybalsley.atlassian.net` instance
4. Connect your `ybotman/tangotiempo.com` repository

Step 2: Verify Smart Commits

Your existing commit format already works:

```bash
TIEMPO-123: Brief description of change

- Detailed point 1
- Detailed point 2

AI-Guild Role: Builder
```

Step 3: Enhanced GitHub Actions Integration

Create `.github/workflows/jira-integration.yml`:

yaml

```yaml
name: JIRA Integration
on:
  push:
    branches: [DEVL, TEST, main]
  pull_request:
    types: [opened, synchronize, closed]
  workflow_run:
    workflows: ["Azure Deploy", "Vercel Deploy"]
    types: [completed]

jobs:
  update-jira:
    runs-on: ubuntu-latest
    steps:
      - name: Extract JIRA ticket from commit
        id: jira
        run: |
          TICKET=$(echo "${{ github.event.head_commit.message }}" | grep -o 'TIEMI
          echo "ticket=$TICKET" >> $GITHUB_OUTPUT
          echo "Found ticket: $TICKET"

      - name: Update JIRA on successful build
        if: steps.jira.outputs.ticket && github.event.workflow_run.conclusion ==
        run: |
          curl -X POST "https://tobybalsley.atlassian.net/rest/api/2/issue/${{ st
            -H "Authorization: Bearer ${{ secrets.JIRA_TOKEN }}" \
            -H "Content-Type: application/json" \
            -d '{
              "body": "✅ **[Automated Update]**\n\nBuild successful on `${{ gith
            }'

      - name: Update JIRA on failed build
        if: steps.jira.outputs.ticket && github.event.workflow_run.conclusion ==
```

```
      run: |
        curl -X POST "https://tobybalsley.atlassian.net/rest/api/2/issue/${{ st
          -H "Authorization: Bearer ${{ secrets.JIRA_TOKEN }}" \
          -H "Content-Type: application/json" \
          -d '{
            "body": "❌ **[Build Failed]**\n\nBuild failed on `${{ github.ref_n
          }'
```

Step 4: Required GitHub Secrets

Add these to your repository secrets:

```
JIRA_TOKEN=<your-jira-api-token>
JIRA_URL=https://tobybalsley.atlassian.net
```

Phase 2: Azure Integration

Option A: Via GitHub Actions (Recommended)

Enhance your existing Azure deployment workflow:

yaml

```yaml
- name: Notify JIRA of Azure deployment
  if: success()
  run: |
    TICKET=$(echo "${{ github.event.head_commit.message }}" | grep -o 'TIEMPO-[0-9
    if [ ! -z "$TICKET" ]; then
      curl -X POST "${{ secrets.JIRA_URL }}/rest/api/2/issue/$TICKET/comment" \
        -H "Authorization: Bearer ${{ secrets.JIRA_TOKEN }}" \
        -H "Content-Type: application/json" \
        -d '{
          "body": "🚀 **[Azure Deployment]**\n\nSuccessfully deployed to Azure.\n
        }'
    fi
```

Option B: Azure DevOps Connector

- Install Azure DevOps connector from Atlassian Marketplace
- Direct integration between Azure and JIRA
- More setup but native integration

Phase 3: Vercel Integration

Method 1: Vercel Webhooks (Recommended)

Create webhook endpoint in your app:

javascript

```javascript
// api/jira-webhook.js
export default async function handler(req, res) {
  if (req.method !== 'POST') {
    return res.status(405).json({ message: 'Method not allowed' });
  }

  const { deployment, project } = req.body;

  // Extract JIRA ticket from git commit or branch name
  const ticket = deployment.meta?.githubCommitMessage?.match(/TIEMPO-\d+/)?.[0];

  if (ticket && deployment.state === 'READY') {
    const deploymentUrl = deployment.url;
    const environment = deployment.target === 'production' ? 'Production' : 'Prev

    await fetch(`${process.env.JIRA_URL}/rest/api/2/issue/${ticket}/comment`, {
      method: 'POST',
      headers: {
        'Authorization': `Bearer ${process.env.JIRA_TOKEN}`,
        'Content-Type': 'application/json',
      },
      body: JSON.stringify({
        body: `🌐 **[Vercel Deployment]**\n\n${environment} deployment ready!\n\n
      })
    });
  }

  res.status(200).json({ message: 'Webhook processed' });
}
```

Method 2: GitHub Actions Integration

yaml

```yaml
- name: Get Vercel deployment URL
  id: vercel
  run: |
    DEPLOYMENT_URL=$(vercel --token=${{ secrets.VERCEL_TOKEN }} --scope=${{ secre
    echo "url=$DEPLOYMENT_URL" >> $GITHUB_OUTPUT

- name: Update JIRA with Vercel URL
  run: |
    TICKET=$(echo "${{ github.event.head_commit.message }}" | grep -o 'TIEMPO-[0-
    if [ ! -z "$TICKET" ]; then
      curl -X POST "${{ secrets.JIRA_URL }}/rest/api/2/issue/$TICKET/comment" \
        -H "Authorization: Bearer ${{ secrets.JIRA_TOKEN }}" \
        -H "Content-Type: application/json" \
        -d '{
          "body": "🌐 **[Vercel Deployment]**\n\nPreview deployment ready!\n\n**U
        }'
    fi
```

Phase 4: What You'll See in JIRA

Development Panel in JIRA Tickets

- **Branches:** Shows your feature branches
- **Commits:** All commits with messages
- **Pull Requests:** Status and reviews
- **Builds:** Success/failure status
- **Deployments:** Links to deployed environments

Automated Comments

- Build status updates
- Deployment confirmations
- Preview URLs for testing
- Error notifications

Status Transitions

You can automate status changes:

```yaml
- name: Transition JIRA to In Review
  if: github.event_name == 'pull_request' && github.event.action == 'opened'
  run: |
    # Auto-transition ticket to "In Review" when PR is created
```

Phase 5: Benefits

Single Source of Truth

- All build/deployment info in JIRA ticket
- No need to check GitHub, Azure, or Vercel separately
- Complete audit trail of development to production

Team Collaboration

- Stakeholders see deployment status without technical access
- QA team gets preview URLs automatically
- Project managers track progress in real-time

Automated Workflow

- Less manual updates to tickets
- Consistent information flow
- Reduced context switching

Implementation Priority

1. **Start with GitHub for JIRA** (easiest, biggest impact)
2. **Add Azure deployment notifications** (via GitHub Actions)
3. **Integrate Vercel webhooks** (for preview URLs)
4. **Fine-tune automations** based on usage

Maintenance

- Monitor webhook delivery rates
- Adjust comment formats based on team feedback
- Add more automation as patterns emerge
- Consider rate limits for API calls