# Atlassian AI Assistant - Installation & Usage Guide

## 🚀 Quick Start for Atlassian Employees

This guide helps you install and test the AI-powered VS Code extension that integrates Jira, Confluence, and AI for intelligent code generation.

**Simple 3-step process**: Install extension → Run command → Follow setup prompts → Start using!

## Prerequisites Checklist

### ✅ What You Need

* Windows, Mac, or Linux computer
* Access to Atlassian Cloud (Jira + Confluence)
* Admin access to create API tokens
* OpenAI API key (for AI code generation)
* **IMPORTANT**: At least one Jira project with **well-written** issues (detailed descriptions, clear requirements)
* **IMPORTANT**: At least one Confluence space with **substantial** documentation (coding standards, architecture docs, examples)
* 10 minutes for setup

⚠️ **Critical for Success**: This extension’s effectiveness depends entirely on the **quality and depth** of your Atlassian content:

**EXCELLENT Results When You Have:** - Jira stories with detailed descriptions, acceptance criteria, and technical requirements - Confluence pages with coding standards, architectural patterns, API documentation, or code examples - Well-organized, team-maintained documentation

**POOR Results When You Have:** - Sparse Jira tickets with minimal descriptions (“Fix bug”, “Add feature”)  
- Empty or outdated Confluence spaces - Little to no technical documentation

**Bottom Line**: The AI can only be as good as your content. If your Atlassian instance lacks detailed stories and rich documentation, the generated code will be generic and unhelpful.

💡 **Want to see what good content looks like?** Check the **“1. CASE STUDY.docx”** to see examples of well-written Jira stories and Confluence documentation that produce excellent AI-generated code.

### ✅ What You’ll Get

* Working VS Code extension that reads Jira stories
* AI code generation based on your team’s Confluence documentation
* Live demo of integrated Atlassian ecosystem + AI
* **No coding required** - automatic configuration setup

### 📁 What’s Included

* **atlassian-ai-assistant-0.0.1.vsix** - ✅ **Install this to use the extension**
* **7. Extension Source Code/** - 📚 **Optional: Technical reference and learning material**

## Step 1: Install Visual Studio Code (if needed)

### If You Already Have VS Code:

✅ Skip to Step 2

### If You Don’t Have VS Code:

1. **Download**: Go to https://code.visualstudio.com/
2. **Install**: Run the installer (accept all defaults)
3. **Launch**: Open VS Code after installation

## Step 2: Install the Atlassian AI Assistant Extension

### Method 1: Install from File (Recommended)

1. **Download**: Get the atlassian-ai-assistant-0.0.1.vsix file
2. **Open VS Code**
3. **Open Command Palette**: Press Ctrl+Shift+P (Windows/Linux) or Cmd+Shift+P (Mac)
4. **Type**: Extensions: Install from VSIX
5. **Select the File**: Browse to and select atlassian-ai-assistant-0.0.1.vsix
6. **Wait**: Extension installs automatically
7. **Success**: You’ll see “Extension installed successfully” message

### Method 2: Manual Installation (Alternative)

1. Open VS Code
2. Go to **View** → **Extensions** (or press Ctrl+Shift+X)
3. Click the **“…”** menu in the Extensions panel
4. Select **“Install from VSIX…”**
5. Choose the atlassian-ai-assistant-0.0.1.vsix file

Test it worked

1. Go back into the **Command Palette**: Press Ctrl+Shift+P, and typing “Atlassian”
2. You will see 2 capabilities:
   1. 🤖 Generate Code from Jira Story
   2. 🚀 Open Your Atlassian Project

## Step 3: Set Up API Access

### Create Atlassian API Token

1. **Go to**: https://id.atlassian.com/manage-profile/security/api-tokens
2. **Click**: “Create API token”
3. **Label**: “VS Code AI Extension Demo”
4. **Copy**: Save the token somewhere safe (you’ll need it in Step 4)

### Get OpenAI API Key

1. **Go to**: https://platform.openai.com/api-keys
2. **Create account** if needed (or sign in)
3. **Click**: “Create new secret key”
4. **Copy**: Save the key somewhere safe (starts with sk-)

### Find Your Atlassian Site URL

* Your Jira URL looks like: https://yourcompany.atlassian.net
* Note the part before .atlassian.net (this is your site name)

## Step 4: First-Time Setup (Automatic Configuration)

### ✅ **Simple & Automatic - No Code Editing Required!**

The extension automatically prompts you for configuration when you first use it. Just install and run - the extension handles everything!

### What Happens on First Use:

1. **Run any command** (🤖 Generate Code or 🚀 Open Project)
2. **Extension detects** you haven’t configured it yet
3. **Guided setup prompts** appear automatically:
   * Enter your Atlassian email
   * Enter your Atlassian site (e.g., yourcompany.atlassian.net)
   * Enter your API token (from Step 3)
   * Enter your OpenAI API key
4. **Configuration saved** automatically to VS Code settings
5. **Ready to use immediately!**

### One-Time Setup Information:

* **Atlassian Email**: The email you use to log into Atlassian
* **Atlassian Site**: Found in your Jira URL (e.g., yourcompany.atlassian.net)
* **API Token**: Created in Step 3 above
* **OpenAI API Key**: Get from https://platform.openai.com/api-keys

### ⚠️ Important Notes:

* **Setup runs once** - extension remembers your configuration
* **All prompts include validation** - ensures correct format
* **Secure storage** - credentials saved in VS Code’s secure settings
* **Can reconfigure** - delete settings in VS Code preferences to reset

### 📁 About the Source Code (Optional)

The **“7. Extension Source Code”** folder is included for technical reference only. You don’t need to touch it to use the extension! It’s provided for: - 🔍 **Transparency** - See exactly how the extension works - 📚 **Learning** - Explore professional VS Code extension development patterns - 🛠️ **Customization** - Modify or extend if desired (advanced users only)

**Just use the .vsix file for the full experience!**

## ⚠️ IMPORTANT: Understanding the Extension Configuration

### Critical Setup Requirement

**The extension must be configured to work with YOUR Atlassian instance.** Since Atlassian sites are private, you cannot access the development environment where this was built. You’ll need to set it up for your own Jira and Confluence.

### What You Need

* Access to Atlassian Cloud (Jira + Confluence)
* Admin permissions to create API tokens
* At least one Jira project with some issues
* At least one Confluence space with some documentation

### If You Don’t Have These

You can still install the extension to see the interface and commands, but the API calls will fail when trying to connect to Jira/Confluence. The extension will demonstrate the concept even without live data.

## Step 5: Test the Extension

### Quick Test Sequence

1. **Open Command Palette**: Ctrl+Shift+P (Windows/Linux) or Cmd+Shift+P (Mac)
2. **Type**: “Atlassian”
3. **You should see**:
   * 🚀 **Open Your Atlassian Project**
   * 🤖 **Generate Code from Jira Story**

### Test 1: Try Any Command (This Triggers Setup)

1. **Run**: Either “🚀 Open Your Atlassian Project” or “🤖 Generate Code from Jira Story”
2. **First time**: Extension will prompt for configuration (see Step 4)
3. **Follow the prompts**: Enter your Atlassian and OpenAI details
4. **Success**: Extension configured and ready to use!

### Test 2: Open Your Atlassian Project

1. **Run**: “🚀 Open Your Atlassian Project”
2. **Choose**: “Open Jira Projects” or “Open Confluence Spaces”
3. **Result**: Your browser opens to your actual Atlassian instance

### Test 3: Code Generation

1. **Create**: New file in VS Code (Ctrl+N)
2. **Run**: “🤖 Generate Code from Jira Story”
3. **Enter**: A Jira issue key from YOUR project (e.g., “PROJ-1”, “TASK-5”, etc.)
4. **Watch**: AI generates TypeScript code in your file
5. **Result**: Professional code appears that combines your Jira story with any Confluence documentation patterns!

**Note**: Use issue keys from YOUR Jira project. The extension will fetch real data from your Atlassian instance.

## Step 6: Understand What You’re Seeing

### The Magic Moment

When you run “Generate Code from Jira Story” with your own issue:

1. **Extension reads** your Jira story details (title, description, requirements)
2. **Extension fetches** any relevant documentation from your Confluence
3. **AI combines** both sources to generate contextually-aware code
4. **Result**: Code that reflects your specific requirements AND any team patterns from your documentation

### Why This Matters

* **Better AI**: Context from organized knowledge beats generic AI responses
* **Team Consistency**: Generated code can follow established patterns if documented in Confluence
* **Developer Productivity**: Less context switching, more focused development
* **Competitive Advantage**: Shows why Atlassian’s integrated platform enables superior AI experiences

### What You’ll See

The generated code will vary based on your actual Jira stories and Confluence content. The more structured information you have in your Atlassian instance, the better the AI-generated results will be.

## Troubleshooting

### Extension Not Appearing?

* **Check**: Extensions panel (Ctrl+Shift+X) for “Atlassian AI Assistant”
* **Try**: Developer: Reload Window (Ctrl+Shift+P)
* **Restart**: Close and reopen VS Code

### Commands Not Working?

* **Verify**: API tokens are correctly configured
* **Check**: Internet connection for API calls
* **Look**: At VS Code Developer Console for error messages

### API Errors?

* **Token expired**: Create new API token
* **Wrong site**: Verify your Atlassian site URL
* **Permissions**: Ensure token has Jira and Confluence read access

### Generated Code is Generic/Poor Quality?

**This is the most common issue!** The extension is working correctly, but your Atlassian content isn’t rich enough:

* **Check your Jira story**: Does it have detailed descriptions, acceptance criteria, technical requirements?
* **Check your Confluence**: Do you have coding standards, examples, architectural docs the AI can reference?
* **Compare to Case Study**: Look at “1. CASE STUDY.docx” for examples of the content quality needed
* **Remember**: Generic input = generic output. The AI needs good source material to generate good code.

## Next Steps

### For Your Own Project

1. **Set up**: Your own Jira project with real stories
2. **Create**: Confluence pages with your team’s coding standards
3. **Configure**: Extension to use your Atlassian instance
4. **Generate**: AI code that follows YOUR team patterns

### Strategic Discussion Points

* How does this change developer productivity?
* What’s the competitive advantage vs. scattered documentation?
* How could this scale across your entire development organization?
* What other AI integrations become possible with organized knowledge?

## 🎯 Success Criteria

### You Know It’s Working When:

✅ Extension commands appear in Command Palette  
✅ “Show Demo” displays working links to Jira/Confluence  
✅ “Generate Code” prompts for issue key  
✅ Generated TypeScript code appears in editor  
✅ Code quality shows clear influence from team standards

### You’re Ready to Demo When:

✅ You can run the full workflow smoothly  
✅ You understand the strategic value proposition  
✅ You can explain why organized knowledge enables better AI  
✅ You can discuss competitive advantages

## 📞 Need Help?

If you run into any issues: 1. **Check** the troubleshooting section above 2. **Look** at the VS Code Developer Console for errors 3. **Try** the demo instance first before configuring your own 4. **Remember**: This is a prototype - focus on the concept demonstration

**The goal is to show the art of the possible, not production perfection!**

## 🚀 Ready to Be Amazed?

This extension represents the future of AI-powered development: intelligent, contextual, and seamlessly integrated into existing workflows.

**Start with Step 1 and enjoy the demo!** 🎉