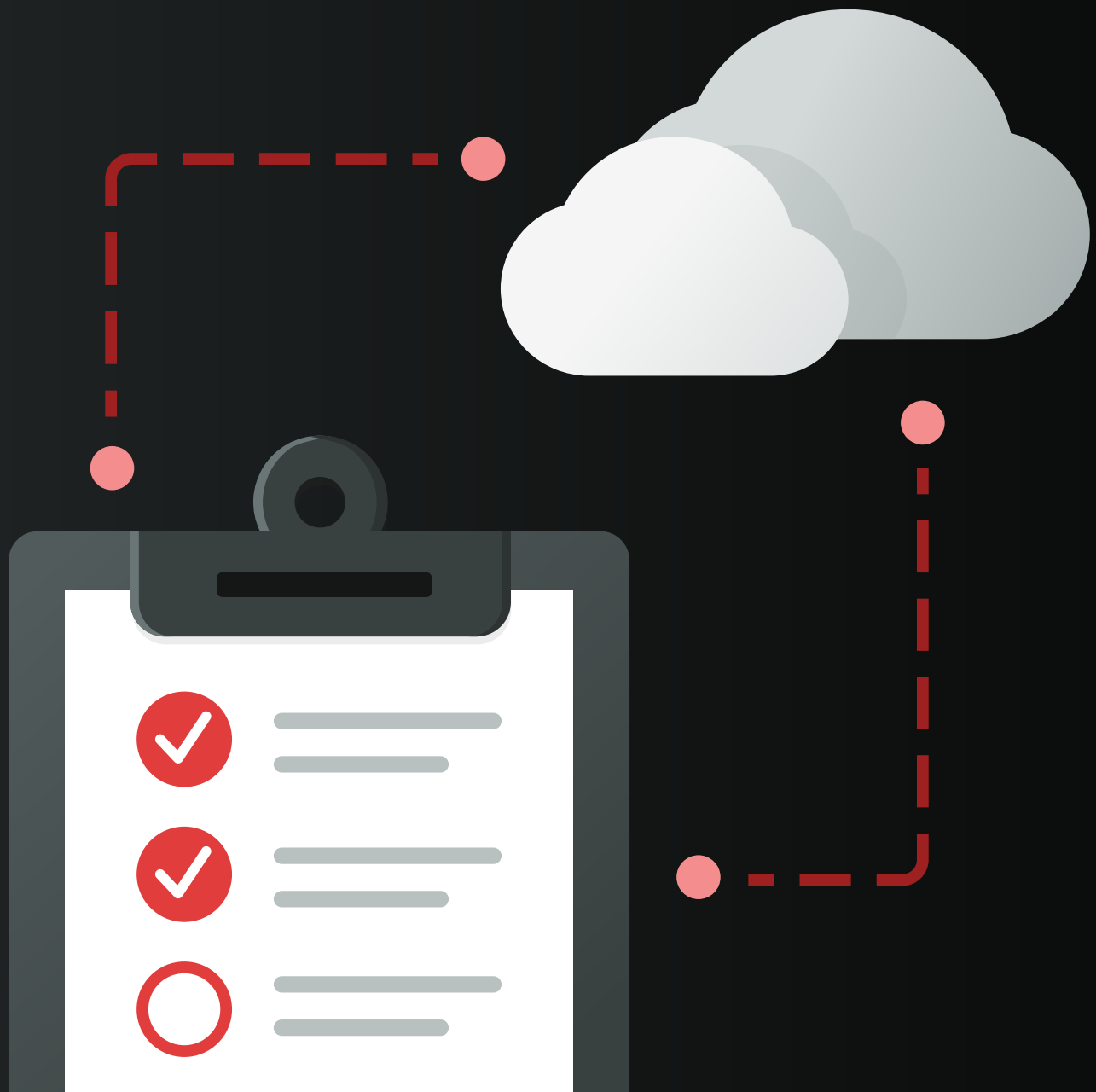
Migrate & Manage

appfire

Jira Migration Checklist



CHEAT SHEET

Jira Migration Checklist

BEFORE PROJECT PREPARATION

- [ ] **1. Determine goals and motivations**
  Why migrate? What are the benefits? What features or efficiencies are gained? What is reduced? What are the risks of not migrating?
- [ ] **2. Research deployment differences**
  What feature, pricing, capability, and app differences exist between Jira Cloud, Server, and Data Center?
- [ ] **3. Determine migration path**
  e.g., Jira Server to Jira Data Center, Jira Server to Jira Server (merge/consolidation), start fresh, etc.
- [ ] **4. Determine initial migration scope**
  Do you plan to migrate all data or some data?
- [ ] **5. Gather source application details**
  What is the source application's URL, deployment type, and version? How many users are there? What are the known problems or challenges?
- [ ] **6. Gather source application stats**
  How many projects, issues, attachments, and custom fields exist?
- [ ] **7. Gather source app and customization details**
  Which apps are installed in the source application?
- [ ] **8. Verify source license expiration date**
  Verify expiration date is valid through the migration period. Visit: Admin > System > License
- [ ] **9. Review migration tool features and limitations**
  To Cloud: Jira Cloud Migration Assistant, Jira Site Import, Configuration Manager for Jira, CSV Import, etc.
  To Data Center: Configuration Manager for Jira, CSV Import, etc.
- [ ] **10. Determine application model (For Jira Data Center only)**
  Will you run Jira as a clustered application or single application?
- [ ] **11. Examine migration scope**
  Are there projects, issues, or settings to exclude?
- [ ] **12. Research long-term storage and retention policy**
  Where can historical data be stored? How long does it need to remain available? In what format?
- [ ] **13. Identify stakeholders**
  Create a list of impacted groups and lead contacts. Record benefits, impacts, concerns, and actions.
- [ ] **14. Identify migration team members**
- [ ] **15. Get stakeholder buy-in**
  Determine whether you need to convince others that migration is needed.
- [ ] **16. Create migration proposal**
  Create a proposal to communicate the migration goals, benefits, and high-level plan.
- [ ] **17. Research approval requirements**
  What approvals are needed? When? By whom?

- [ ] **18. Get preliminary project approval**

  Make sure leadership is aware of the project and agrees on the overall execution strategy.

- [ ] **19. Create a project tracking issue in Jira**

  Use Jira to track Jira support.

- [ ] **20. Create a project documentation space in Confluence**

  Create a companion Confluence space to document Jira support. Link the Confluence space to the Jira project.

- [ ] **21. Setup regular project meetings**

  Create a regular project schedule to share updates, discuss issues, and make decisions.

- [ ] **22. Create a decisions list**
- [ ] **23. Conduct source configuration assessment**

APPLICATION PREPARATION

- [ ] **24. Upgrade source Jira and apps**
- [ ] **25. Evaluate app compatibility**

  Are all apps needed in the target application? Are all compatible with the target application? What functional differences exist?

- [ ] **26. Evaluate connections, integrations, and customizations**

  Are all connections and integrations needed in the target application? Are all compatible with the target application? What customizations need additional research or special treatment? Do customizations work in a clustered environment (e.g., Jira Data Center)?

- [ ] **27. Back up source application**

  Develop and test a backup strategy. Back up the source application.

- [ ] **28. Archive unneeded source projects**

  Archive projects or export unneeded issues for long-term storage.

- [ ] **29. Clean up source application**

  Remove any unused schemes or apps. Reduce duplication and unneeded settings.

- [ ] **30. Run source health checks (For Jira Server and Jira Data Center only)**

  Resolve any issues reported by the following tools:

  Instance Health Tool: (Admin > System > Troubleshooting and support tools)

  Integrity Checker: (Admin > System > Integrity checker)

- [ ] **31. Set up target application**

  Install and configure the target application. Verify that system settings (e.g., mode, default language, user locale, default user time zone, etc.) match the source application. Verify user limit is adequate. Verify the license expiration date is valid through the migration period. Visit: Admin > System >License

- [ ] **32. Gather target application details**

  What is the target application's URL, deployment type, and version?

- [ ] **33. Conduct target configuration assessment (for merge/consolidation)**
- [ ] **34. Set up staging application**

  Install a staging application using the same settings and specs as the target application.

- [ ] **35. Gather staging application details**

  What is the staging application's URL, deployment type, and version?

- [ ] **36. Run source performance tests (Jira Server and Jira Data Center)**
  Run performance tests to ensure the source, staging, and target applications can handle the import/export process, additional data, and new user load. Increase system resources (e.g., computing memory, JVM memory, etc.) In the target application to handle additional data and user requests. Add a dedicated node in Jira Data Center.
- [ ] **37. Determine helper apps**
  Determine which apps are needed for the migration. Install apps on other applications as needed.
- [ ] **38. Check source object integrity**
  Use the Integrity Check for Jira app to detect and address filter, dashboard, and board errors.

MIGRATION PREPARATION

- [ ] **39. Determine user account strategy**
- [ ] **40. Determine migration execution strategy**
  Decide whether to enlist help from Atlassian, a Solution Partner, or migrate on your own.
- [ ] **41. Create vendor access**
  Create application, server, and database access accounts for any migration vendors.
- [ ] **42. Determine timeline**
- [ ] **43. Build a communication plan**
- [ ] **44. Make final app and customization decisions**
- [ ] **45. Address and resolve conflicts**
  Identify and discuss how to handle conflicts like duplicate project keys, naming collisions, similar schemes, etc.
- [ ] **46. Form a migration test team**
- [ ] **47. Create data test cases**
- [ ] **48. Create test accounts**
- [ ] **49. Create functional test cases**

MIGRATION TESTING

- [ ] **50. Create rollback plan**
  Create, verify, and document a plan to handle an emergency rollback.
- [ ] **51. Develop launch plan**
- [ ] **52. Refresh environment data**
  Update the staging environment configuration and data if needed.
- [ ] **53. Select and communicate configuration freeze window**
- [ ] **54. Test launch plan and migration**
  Test the entire migration process in a staging environment.
- [ ] **55. Update staging settings**
  Update the base URL, application links, property files, and any other settings as needed.
- [ ] **56. Execute test cases**
  Verify that all data was transferred and functions as expected. Check the application logs for any unexpected issues.

- [ ] **57. Address issues**
  Fix any issues encountered during the test migration process.
- [ ] **58. Repeat test migration as necessary**
  Do as many trial runs as needed. Don't progress from the staging environment until everything is perfect.

ADDITIONAL PLANNING

- [ ] **59. Refine launch plan**
  Update the launch plan to incorporate findings from the test migration.
- [ ] **60. Select migration date**
  Use the "Selecting a Launch Window" information in the "Migration Considerations" section to select a migration date.
- [ ] **61. Communicate launch plan**
  Communicate plans to stakeholders and users.
- [ ] **62. Conduct pre-migration training**

DURING

- [ ] **63. Execute launch plan**
  Execute the steps in the launch plan previously developed, tested, and refined.

AFTER

- [ ] **64. Perform post-migration clean-up**
  Address any unneeded or duplicate settings or schemes.
- [ ] **65. Decommission source application**
  Start the decommission process. Prevent application changes, make data "read only", redirect visitors, disable login ability, and finally, take the source application offline.
- [ ] **66. Maintain staging environment**
  Keep the staging environment's configuration in sync with production.
- [ ] **67. Create project templates**
  Prevent over-customization by creating templates for development, support, and task projects.
- [ ] **68. Revisit disaster recovery and backup strategy**
  Make sure the target application is regularly backed up. Use Configuration Manager for Jira to take periodic snapshots and quickly deploy the "last known good" configuration.
- [ ] **69. Revisit archive strategy**
  Periodically archive any unneeded projects and issues in the target application.
- [ ] **70. Conduct post-migration training**
- [ ] **71. Mark tasks complete**
  Mark Jira tracking tasks complete and update project documentation as needed. Log any outstanding items to address.
- [ ] **72. Conduct a retrospective**
  Document any issues encountered, what went well, and what to improve for future projects.

- [ ] **73. Thank upgrade team**

  Recognize the efforts of everyone who participated in the project.

This worksheet is from **The Essential Workbook for Planning Jira Migrations** by Rachel Wright.
Sign up to get the full guide once it launches: https://appf.re/migrations-workbook