



Long-term agile planning with

JIRA Software and Portfolio for JIRA

Authors



Robin Scanlon

Robin is a Portfolio for JIRA buff and Atlassian Expert at GLINTECH. He's been consulting and training people in the software world for 10 years, particularly in the project management space. Originally from Scotland, he now calls Sydney home, and when not at work can be found running or drinking wine.



Rhys Christian

Rhys is on the Portfolio for JIRA product management team. Only in his second year in the industry he's delighted by delivering great software. A born and raised Sydneysider, he loves technology, skiing, rugby and a nice Scotch.



Laura Daly

Laura is an agile guru at Atlassian with experience on various product teams including JIRA Software, Portfolio for JIRA, and Bitbucket. When she is not writing about agile best practices you can find her in the mountains chasing storms or looking for the perfect berm.

Contents

Introduction **4**

The iron triangle of planning **6**

How far out to plan **8**

Planning concepts in Portfolio for JIRA **10**

Getting started with Portfolio for JIRA **14**

Work with your plan **28**

Take-aways **37**



Introduction

The benefits of agile software development, like high-quality code and faster release cycles, are well known. But agile methodologies can be mistaken as an ‘ad-hoc’ approach to working; most agile teams limit planning to the immediate future, like what are we doing in the next sprint?

The truth is that when using an agile approach, there is still the need to forecast over a long-time period. Without a long-term plan, software teams cannot align a roadmap to strategic business objectives and be successful.



Some teams try to track long-term plans and strategic goals with disconnected tools like spreadsheets, but it is not easy. To avoid this painful and time-consuming approach, Atlassian provides a seamless way to plan in [JIRA Software](#) with [Portfolio for JIRA](#). This white paper will take a dive into how to tackle agile roadmap planning and how to set up Portfolio for JIRA and JIRA Software for your teams.



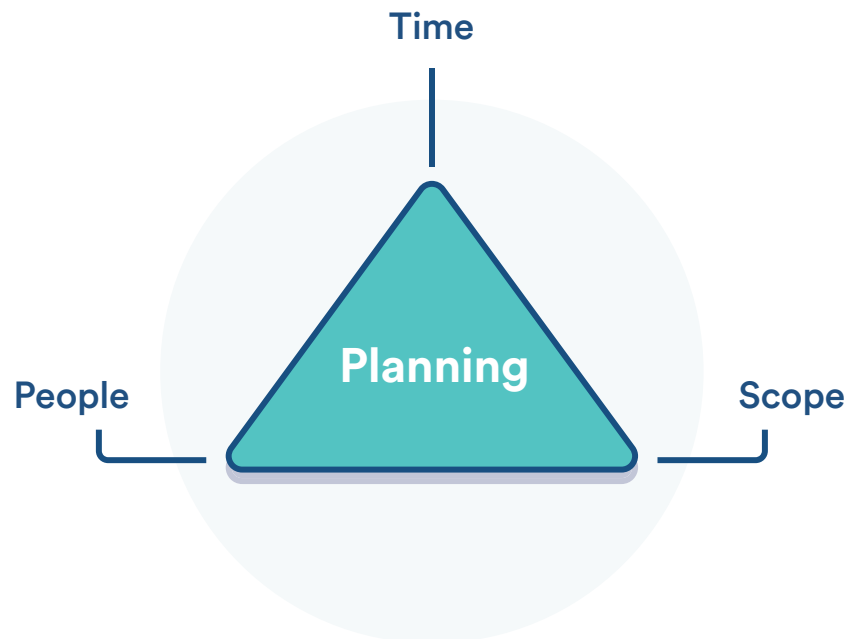


The iron triangle of planning

All agile software projects have goals: what the project needs to deliver, when it needs to be delivered by, and within what budget. However, managing these three constraints can be a complex juggling act.

So let's take a cue from the iron triangle of planning and learn how balancing different variables can help agile software teams plan effectively. The three variables are:

- Scope – what is the work we are planning for? Without this we can all go home!
- Resources – who is going to do the work?
- Time – how long will this take?



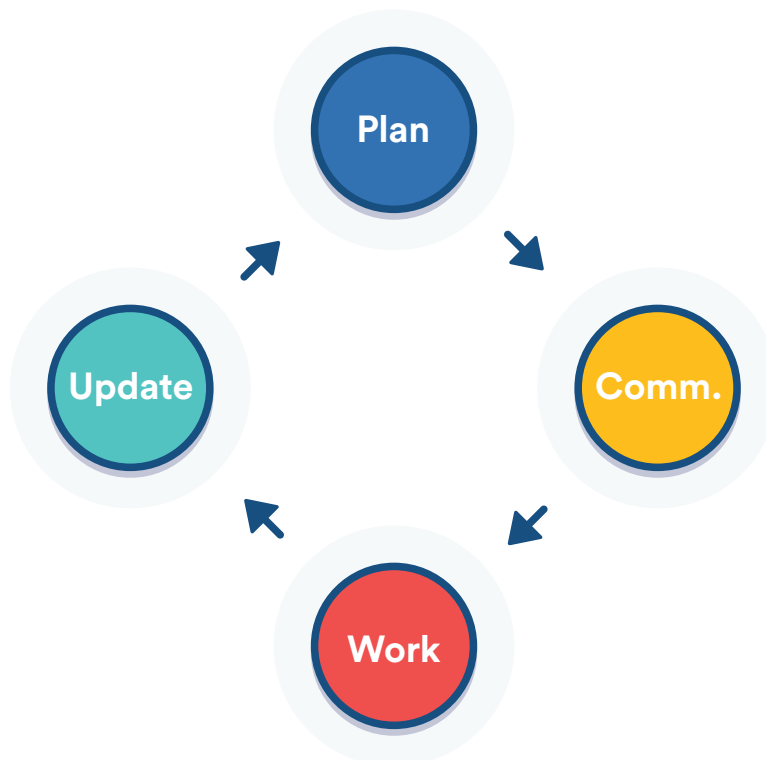
Each of these elements are linked to the other; if you increase the scope, you need more time, or more people to accommodate it. Or if your deadline is moved and you have less time, you need to reduce the amount of work or add more people to balance it out.

Planning is the art of finding the best combination of these three things to satisfy the most requirements. If your team understands where their work sits in the bigger picture, it helps focus attention on the tasks that really matter.

 **Pro tip**

Plans are not static; variables change in an agile environment. Feeding this information back into your plan and re-planning is essential along with communication. There is no point to having a plan if no one knows what it is.

Agile is all about empowering teams and it's essential to include all members of the team in the planning process at some point. Otherwise you end up in a command-and-control environment, prescribing who does what and when.





How far out to plan

Roadmaps provide guidance and direction from six months to five years out depending on the team, product type, and company. This vision usually starts with granular details for the immediate future, like specific user stories that roll up into a feature, and get more fuzzy as the timeline progresses. But you have to start somewhere, and capturing this data in a plan is the first step.

So in this respect, long-term planning is not very different from short-term planning – the same logic applies:

1. Specify the target dates that you aim to complete certain chunks of work (aka epics in JIRA Software).
2. Apply high-level estimates to your chunks of work.
3. As tasks become more defined, break down chunks of works into stories with estimates.
4. Rinse and repeat.

If you are new to long-term planning and roadmapping, a good place to start this process is with a regular big-picture planning meeting. This session can be yearly, quarterly, or monthly; it depends entirely on your business model. We often find this aligns with when business results are due.

These meetings are a time for stakeholders like product owners, product managers, and development managers to get together and plan out big deliverables. The goals of these meetings are to define strategic themes (focus areas) for the roadmap by asking general questions like “where are we going?” and “where should we be spending our time?” Techniques for creating a high-level strategy involve reviewing customer feedback, advancements in the market, and the goals of the wider organization. By the end of this meeting, the team should be on the same page of what you want to ship and when you want to ship it.



After these planning meetings your team will have a concrete roadmap to look back to at the end of each sprint. And it is the job of the roadmap owner – usually a product owner or product manager – to re-baseline the plan after each sprint to adapt to any changes. Luckily with experience, long-term planning gets easier, meaning your forecast is more likely to come true.



Pro tip

Managing a roadmap is not a discrete activity; it should be integral to your existing agile ceremonies. For example, backlog grooming involves prioritizing your work into now, next, and later. Do your latest changes still align with the longer-term plan? If not, update it. Did a blocker get mentioned in your daily standup? Take note. At your retrospective, what was different between your initial plan and the actual outcome? Why?

“If you are new to long-term planning and roadmapping, a good place to start this process is with a regular big picture planning meeting.”



Planning concepts in Portfolio for JIRA

Creating a vision for a product in the form of a roadmap or a list of big-ticket items is a great foundation for starting to understand how Portfolio for JIRA works. These strategic business goals and the understanding of how planning works with scope, resources, and time all come together in Portfolio for JIRA. But before you get started with Portfolio for JIRA, there are a few other considerations you need to make.

Think about your work item hierarchy

Epics in JIRA Software denote large chunks of work that need to get done like a feature or even part of a big feature. When you work with a long-term roadmap and strategic business goals, sometimes it is not enough to have epics as business goals. For example, if you are building a travel mobile app, you might have an epic and stories like:

Epic: Share trip on social

- a. Story: As a user I can share my upcoming trip on Instagram
- b. Story: As a user I can share my upcoming trip on Facebook
- c. Story: As a user I can share my upcoming trip via email

When planning long-term, this epic “Share trip on social” may actually roll up into a bigger business initiative like “Integrations,” which also encompasses integrating with other payment providers.



In the same way that stories are grouped into epics in JIRA Software, you can also group epics into larger pieces of work for our roadmap like, **initiative > feature > epic > story**.

Initiative: Integrations

- a. Epic: Share trip on social
 - Story: As a user I can share my upcoming trip on Instagram



Pro tip

What other information do you need from your plan? Reporting needs at the portfolio level will differ from project- and team-level metrics. Fortunately, there are a number of portfolio-specific reports that address these new requirements and levels of hierarchy.

Pick a planning method

Looking back at the tenants of agile planning, releases are the time part of the equation. When you begin a Portfolio for JIRA plan* one of the first things to consider is, what kind of planning is best for your teams? Working towards a fixed date and changing the scope? Or keeping the scope and changing the release date?

Portfolio for JIRA supports both time-boxed and scope-boxed planning and you do not need to choose a single approach, because this can differ for each release. The differences between the two planning methods are as follows:

- **Time-boxed planning** – this is where your release date is set and cannot be moved. The release is nailed into your timeline and Portfolio for JIRA will show you if you are forecasted to meet that date (timeline in green) or if you are likely to miss it (timeline in red).

**A plan in Portfolio for JIRA is a visual roadmap where you can play with creating reliable forecasts, staying informed with realistic schedules, and effectively troubleshoot and manage releases in an ever-changing environment. You can create any number of plans which can be accessed by different users and groups.*



- **Scope-boxed planning** – this is when your releases have a dynamic release date allowing Portfolio for JIRA to forecast the proposed date based on the scope of work you assign to the release, and the number of people who can work on it. This method is better for indicative planning or MVP releases, where the date is dictated by certain scope completion.

Since work items change over time – especially estimates – there is another option for teams that do not know release dates and scope assignments when creating a roadmap. Portfolio for JIRA caters to this use-case by adding a later release to each project. This acts as a ‘catch all’ bucket in the plan for items or assignments that have yet to be scoped. When you are ready to update your plan, you can scoop issues out of the ‘later’ release to a more specific one as necessary. The later release will start after the last defined release and extend to when all remaining work is forecast to end.

Use teams as your resource

In the same way that releases are our time element in the plan, teams are our resources. This gives roadmap owners the ability to play with scope and resources in the same tool.

There are two approaches to managing teams in Portfolio for JIRA:

- **Manage resourcing at the team level** – here you specify the number of teams you have, what boards or projects they are assigned to, and what velocity they get through per sprint (or how many hours a week for Kanban).

In this approach, epics and stories are assigned to teams, but the plan is not prescriptive about which member of the team must work on each task. This allows for the teams to self-manage their work. Using the team velocity to estimate effort will account for all the meetings and other distractions that crop up during work time.



- Assign resources individually – you can identify the teams and board assignments as above, but additionally add the names of the team members. Portfolio for JIRA will then assign these individuals to specific tasks.

This approach is ideal if you have feature teams, where each person has different strengths. By creating **skills** in Portfolio for JIRA and flagging a task as requiring certain skills, you can ensure the most suitable person is assigned to each task.



Pro tip

Users often want to use this great functionality from day one, but avoid running before you can walk; adding skills to the scheduling algorithm adds a layer of complexity that those new to Portfolio for JIRA might find tricky to navigate. Start with team-level resourcing and then, take your training wheels off, and progress to skills-based planning.

Look beyond scope, time, and resources

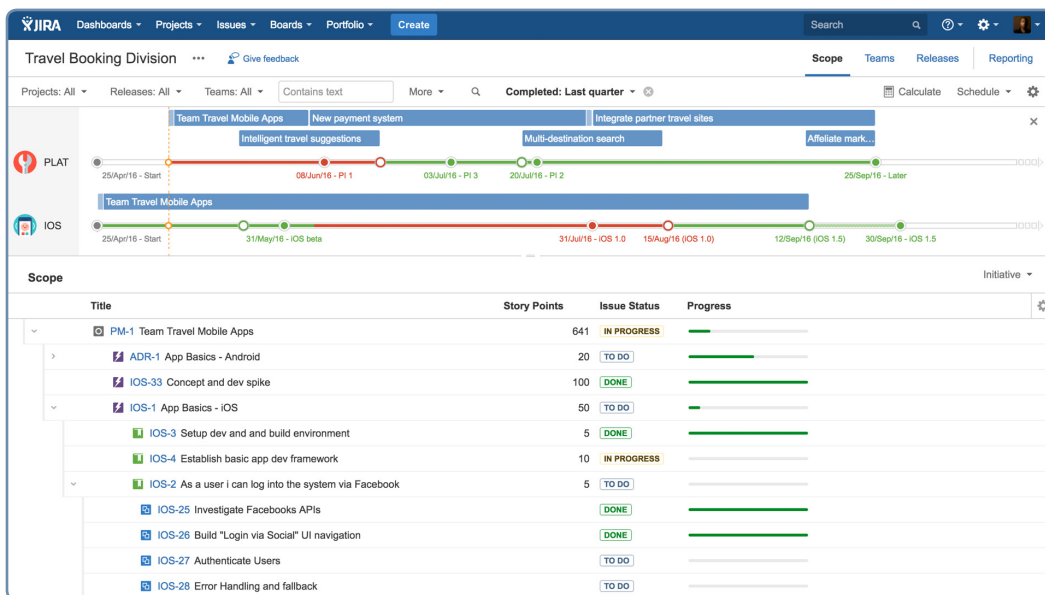
Lastly, communicate. Socialize the plan so everyone knows what part they play in the context of the big picture. Have roadmap check-points in your backlog grooming, or sprint planning sessions to keep your plan visible. Understanding the knock-on effect your deliverables have on the rest of the business can help rally the team. And you will find that getting teams to work together towards a shared goal pays dividends in the long run.

“Lastly, communicate. Socialize the plan so everyone knows what part their work plays in the context of the big picture.”



Getting started with Portfolio for JIRA

Once you start to understand the fundamentals of long-term agile planning and how these concepts work in Portfolio for JIRA, you can begin to build out your Portfolio for JIRA plan.



To help bring everything together, we outlined below how to set up Portfolio for JIRA with your existing JIRA Software instance*. This process will follow the best approach for implementing Portfolio for JIRA:

- Organize your existing JIRA Software projects.
- Configure Portfolio for JIRA.
- Create your Plan.

*The setup is based off of a scrum team methodology, with two-week sprints, and story-point estimation.



1. Organize your issue sources (projects, boards, or filters)

When you create a Portfolio for JIRA plan, you need to select the issue sources you want to pull into your plan. Issue sources are projects, boards, or filters that contain the issues you will use to forecast a roadmap for your plan. Since this setup practices a scrum methodology, we recommend using boards, because they provide the most functionality for managing with Portfolio for JIRA (particularly for managing sprints).

✓ Pro tip

Be careful not to make Fix Version a 'hidden' field as this is a MUST HAVE for Portfolio for JIRA. Also, be careful when setting up 'required' custom fields as they can prevent issues from being created inside Portfolio for JIRA.

If you plan to use a custom hierarchy such as initiatives we recommend you create a separate project and store those issues independently. In that project configure the **issue types** and, if you do not already have one, create an issue type for your custom hierarchy issue type. *If you're not using a custom hierarchy, you can skip to step 3.*

✓ Pro tip

Not sure about hierarchy? Don't worry, you can revisit them later and just work with epics and stories initially.

The screenshot shows the JIRA Project settings page. The top navigation bar includes 'Dashboards', 'Projects', 'Issues', 'Boards', 'Portfolio', and a 'Create' button. The main content area is titled 'Project settings' and is divided into two columns. The left column contains a sidebar with icons for 'Summary', 'Details', 'Re-index project', 'Delete project', 'Issue types', and 'Epic'. The right column is titled 'Issue types' and contains the following text: 'Keep track of different types of issues, such as bugs or tasks. Each issue type can be configured differently.' Below this, it says 'Scheme: PM: Kanban Issue Type Scheme'. At the bottom, there are two issue types listed: 'Epic' and 'Initiative'.



2. Configure a custom hierarchy (optional)

Now that you have a project and issue type for your custom hierarchy set up, you will need to configure Portfolio for JIRA so that it knows what issue type to link to the hierarchy level.



1. Go to **Administration > Add-ons > Portfolio Hierarchy Configuration**
2. **Create Level** and map them to your JIRA issue type's



Pro tip

Do not worry if your hierarchy requirements change - you can edit these levels any time and Portfolio for JIRA will reconfigure your roadmap accordingly - nothing is set in stone!

This useful feature aggregates the story points (or estimation) from sub-task all the way up to the uppermost level – giving you accurate estimates for large, cross-team, cross-project pieces of work – all of which is essential in creating a useful roadmap.

#	Level Name		JIRA Issue Types	Actions
1	Initiative	←	 Initiative	Remove
	Epic	←	 Epic	
	Story	←	All other standard issue types	
	Sub-Task	←	All sub-task issue types	
	+ Create Level			



Surface custom hierarchy on JIRA tickets

One of the key components of portfolio management - and indeed any agile process - is visibility; it is essential to have end-to-end traceability from board strategy to team tasks. Portfolio for JIRA has a 1:1 relationship with native JIRA Software, meaning these issue types can be viewed in the normal JIRA issue view with their hierarchical parent and child links.

When you decide on your hierarchy, you need to configure this in the Portfolio for JIRA Administration settings. Since this is a global setting it is important to have all the teams using the hierarchy feature in Portfolio for JIRA aligned.

To see this relationship in your JIRA Software project, just add the parent link field to your screens to expose this link and provide full top-down (and bottom-up) visibility.

1. **Administration > Issues > Screens**
2. Edit the screen you wish to add the field to
3. Add the field **Parent-link**



Pro tip

The parent link field can be searched using JQL, so you can create a filter for Kanban boards that show the child issues of certain features/initiatives. If you want to get the list of child issues from a parent issues, type: **“Parent Link” = EX-000**

To get the child issues from multiple parent links, type:
“Parent Link” in (EX-000, EX-001, EX-002, EX-003)



3. Create a plan

Now that your issue sources and hierarchy are in order, it is time to create a plan. *Note: This step along with the steps above are typically done by a product owner or product manager who ultimately has ownership of the roadmap.*

To create a plan you first need to select the Portfolio tab in the header of JIRA Software. Then follow these steps:

1. Click **Create Plan**.
2. Enter a **name** for the plan.
3. Select the scope of work you are planning for. Remember that you can connect to multiple sources to create cross-team, multi-project plans.

There are three types of issue sources:

- **Board** – all issues on that board are included
- **Project** – pulls all issues in that project
- **Filter** – for custom issue selection using JQL

The screenshot shows the JIRA interface for creating a plan. The top navigation bar includes 'Dashboards', 'Projects', 'Issues', 'Boards', 'More', and 'Create'. The main heading is '2 Connect to your teams' work'. Below this, there are three buttons: 'Cancel', 'Back', and 'Next'. The main content area is titled 'Connect to existing boards or projects to start planning and reporting.' It features a list of selected boards: 'Board - IOS app', 'Board - Android App Development', and 'Board - Performance Engineering'. A fourth board is being selected from a dropdown menu, with 'Portfolio Management' highlighted. The dropdown menu also shows other options: 'Scrum: Teams in Space', 'Kanban: TIS Partners Funnel', 'Platform Development', 'IOS app', 'Android App Development', 'Performance Engineering', and 'Team Travel Web'. To the right of the board selection area, there is a diagram showing four interconnected board icons.



Pro tip

If your project has scrum boards linked to it, use that board as your source. Portfolio for JIRA can pull extra metrics like team velocity and sprint details that the project alone does not provide. Just make sure that your boards filter includes all of the issues you want included in the plan.

4. Next select the relevant **Releases** you want to work with. These map to the fix versions from your issue sources.
5. Then you will be prompted with **Teams** where you can add teams. By default, Portfolio for JIRA creates a team per issue source. (These can be amended at any time once the plan has been created.)
6. Finally, **Confirm** the issues that you wish to import – you can toggle the view between epics and stories. If an epic is selected than every issue associated with that epic will be imported. If you do not want to import specific epics or stories, de-select the box next to them.

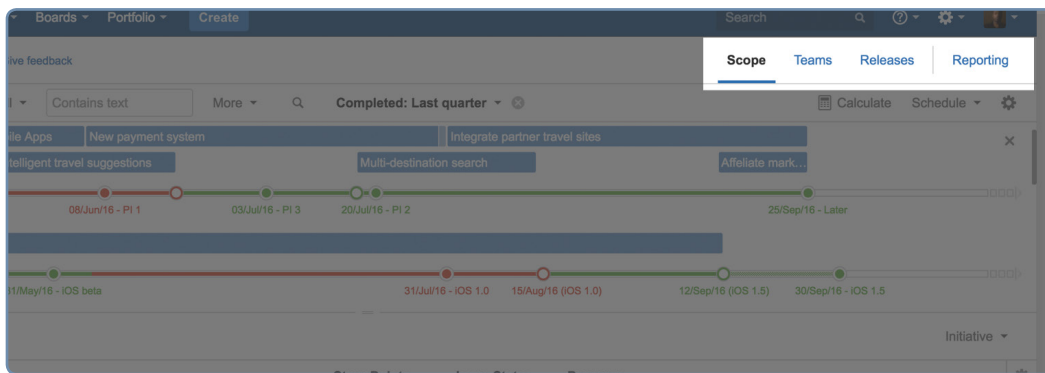
The screenshot shows the JIRA Portfolio interface for the 'Confirm what's in scope' step. The table below lists the issues being reviewed for import.

Title	Releases	Issue Status
<input checked="" type="checkbox"/> IOS-1 App Basics - IOS	IOS beta	TO DO
<input checked="" type="checkbox"/> IOS-9 Trip management	IOS beta	TO DO
<input checked="" type="checkbox"/> IOS-20 Group booking experience	IOS 1.0	TO DO
<input checked="" type="checkbox"/> ADR-1 App Basics - Android	Android beta	TO DO
<input checked="" type="checkbox"/> ADR-16 Basic trip booking	Android beta	TO DO
<input checked="" type="checkbox"/> ADR-15 My Group Trips Overview	Android 1.0	TO DO
<input checked="" type="checkbox"/> ADR-19 Adapt apps to new payment methods		TO DO
<input checked="" type="checkbox"/> ADR-8 Invite and share	Android 1.0	TO DO
<input checked="" type="checkbox"/> ADR-18 Notifications	Android 1.0	TO DO
<input checked="" type="checkbox"/> ADR-17 Collaborative Trip booking flow	Android 1.0	TO DO
<input checked="" type="checkbox"/> PLAT-1 New API Access Points (mobile optimized)	PI 1	TO DO
<input checked="" type="checkbox"/> PLAT-2 Experiments framework for travel suggestions	PI 1	TO DO
<input checked="" type="checkbox"/> PLAT-3 Research and evaluate payment providers	PI 2, PI 3	TO DO
<input checked="" type="checkbox"/> PLAT-4 MVP Integration with selected payments provider	PI 3, PI 4	TO DO
<input checked="" type="checkbox"/> PLAT-5 Full-fledged integration with new payments prov...	PI 4	TO DO
<input checked="" type="checkbox"/> PLAT-6 Billing systems integration - backend		TO DO



Your roadmap at this stage will be produced based on your issue priorities, team velocities, issue estimates, and any fix versions you have sourced into your plan. Note that if your issues do not have estimates, you can create a plan, but the schedule – which is the visual roadmap – will not appear on your plan until your issues and/or epics have estimates.

You will see in the upper right-hand corner the three main areas of a plan that you can manage: **Scope**, **Teams**, and **Releases**. The reporting section provides functionality to present your information in a more digestible view as well as provide visibility to others.





4. Configure your team

Once you have your plan created it is important to go into the team section to make sure that your team configuration – think velocity and capacity metrics – are set up properly. *If you are happy with team configurations set in the plan creation wizard you can ignore this step and go to Step 5.*

Team velocity:

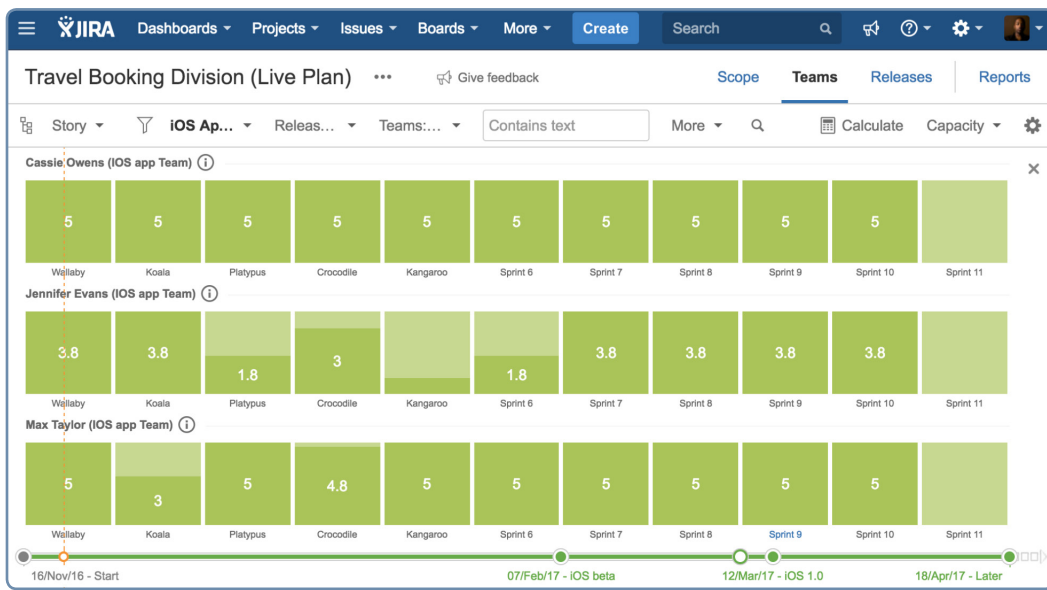
- For **Scrum** boards, Portfolio for JIRA can work out your expected sprint velocity based on the average of your past sprints. This can be changed manually at any time to account for team changes (when you are first getting started with Portfolio for JIRA, this does not need to be adjusted).
- For **Kanban**, set the average hours of work per week that the team gets through. Remember, this is an average number of hours of work completed, which is not the same as the number of hours the team is at work!

The screenshot displays the Jira Portfolio for JIRA interface. At the top, there is a navigation bar with 'Dashboards', 'Projects', 'Issues', 'Boards', 'Portfolio', and 'Create' buttons. Below this, the 'FY17 Plan' is visible, showing a timeline with various tasks and releases. The 'Teams' section is expanded, showing two teams: 'Android App Development Team' and 'IOS app Team'. Each team has a 'Scrum' board type and a velocity value (30 and 35 respectively). The velocity is set to '2 week iteration length' and 'Velocity (pts)'.

Team Name	Board	Methodology	Velocity (pts)
Android App Development Team	Android App Development	Scrum	30
IOS app Team	IOS app	Scrum	35



Once your team velocity is set, you have the option to add team members. You can choose to manage your people at either the team or individual level. Planning from an individual level lets you provide additional variables about availabilities and skills, so our algorithm can calculate a more accurate roadmap. Additionally, while the **Capacity view** is a great feature for development team leads to consider how much work is on the team’s plate, planning from an individual level allows you to dig much deeper into each individual’s capacity.



Pro tip

If you have a member on more than one team, then add that individual to all relevant teams and distribute their weekly hours to proportion their effort between teams. Also note that adding team members is not necessary since you can easily plan with team velocity.



To add team members simply click + **Add person**. You can also create virtual team members to account for variables such as potential new hires.

Teams + Create team

People 2 week iteration length Velocity (pts)

Name	Weekly hours
Mitch Davis	40
Kevin Ca...	40
Ryan Lee	40
Emma Paris	40

+ Add person

Mitch Davis Manage

Global availability

Start date 01/Nov/16
End date 31/Oct/17

Absences

21/Dec/16 – 31/Dec/16 Holiday

Team-specific availability

Android App Development Team

Start date DD/MM/YY
Fill in if joining team after certain date



Pro tip

Make your team a shared team, so you can use them in different plans and you only have to update team member skills once.

Add the **Teams** field to your issue screens to surface the assigned team on the issue.

JIRA Dashboards Projects Issues Boards Portfolio Create

IOS App / IOS-3

Setup dev and build environment

Edit Comment Assign More To Do In Progress Done Admin

Details

Type: Story Status: **TO DO** (View Workflow)
Priority: Major Resolution: Unresolved
Affects Version/s: None Fix Version/s: IOS beta
Labels: None
Epic Link: App Basics - IOS
Sprint: Wallaby

People

Assignee: Robin Scanlon
Reporter: System Admin
Team: IOS app Team
Votes: Vote for this issue
Watchers: Start watching this issue

Dates

Created: 12/Jul/16 5:13 PM
Updated: Just now

Attachments

Drop files to attach, or browse.



Pro tip

Consider using the Team field in other ways – as a quick filter on your epic level Scrum/Kanban board perhaps, to quickly show the workload across the business.

5. Configure your releases

Understanding how teams work in Portfolio for JIRA is important to understanding how velocity is calculated for releases. After that is sorted out, the next step is to decide how you want releases to be mapped out and calculated in the roadmap view of the plan. The release settings will dictate how Portfolio for JIRA schedules tasks in sequence and presents your roadmap as either on-track or overbooked.

If the release has a **Fixed Release Date** then that release is **time-boxed**. Overbooking, assigning too much work to a release, will result in a red timeline:

The screenshot shows the JIRA Portfolio interface for the 'FY17 Plan'. The top navigation bar includes 'Dashboards', 'Projects', 'Issues', 'Tempo', 'Boards', 'Portfolio', and 'Create'. The main area displays a roadmap with several epics: 'iOS-1 App Basics - iOS', 'iOS-9 Trip management', 'iOS this other thing', 'iOS-41 another unestimated epic', 'ADR-8 Invite and share', 'iOS-35 iOS App update', 'iOS-42 new thing we're doing', 'iOS-21 Team invitations', 'iOS-30 Distributed Energy', 'iOS-31 Tech Enablement', and 'iOS-32 Business Capability'. A timeline at the bottom shows dates: 02/Nov/16 - Start, 27/Dec/16 (iOS beta), 31/Jan/17 - iOS 1.0, and 11/Jul/17 - Later. Below the roadmap, the 'Releases' section shows 'iOS App (iOS)' with three release options: 'iOS beta' (02/Nov/16 - 27/Dec/16, marked as OVERBOOKED), 'iOS 1.0' (01/Dec/16 - 31/Jan/17), and 'Later' (02/Nov/16 - 11/Jul/17). The 'iOS beta' release settings are expanded, showing 'Start date' as 'As early as possible', 'Calculated' as '02/Nov/16', and 'Release date' as 'Fixed release date'.

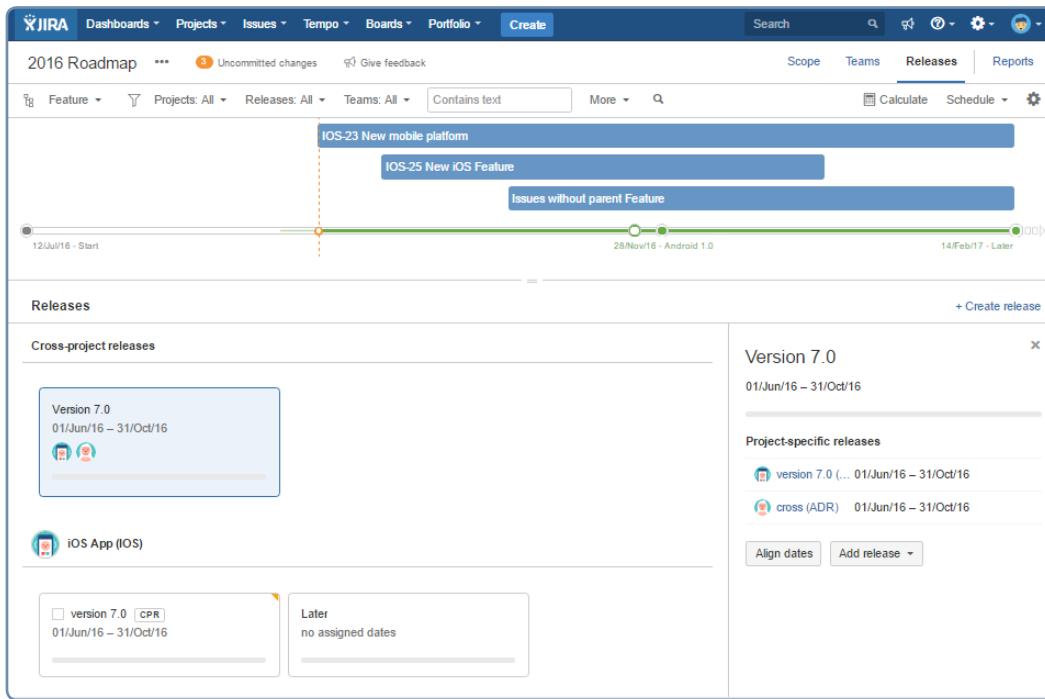


If the release has a **Dynamic Release Date** then that release is using **scope-boxed**. It is not possible to overbook that release – the timeline will just update depending on the scope and resources assigned to that release:

The screenshot displays the JIRA release planning interface. At the top, the 'FY17 Plan' is visible with navigation options for Scope, Teams, Releases, and Reports. The main area shows a Gantt chart for the 'iOS App (iOS)' epic, with tasks such as 'ADR-8 Invite and share', 'iOS-1 App Basics - iOS', 'iOS-9 Trip management', 'iOS-41 another unestimated epic', 'iOS-37 Energy Dashboard', 'iOS-40 new unestimated feature', 'iOS-21 Team invitations', 'iOS-30 Distributed Energy', 'iOS-32 Business Capability', 'iOS-35 iOS App update', 'iOS-31 Tech Enablement', 'iOS-42 new thing we're doing', and 'iOS-20 Group booking experience'. The timeline spans from 02/Nov/16 to 11/Jul/17. Below the chart, the 'Releases' section shows three release options: 'iOS beta' (02/Nov/16 - 27/Dec/16), 'iOS 1.0' (01/Dec/16 - 31/Jan/17), and 'Later' (28/Dec/16 - 11/Jul/17). The 'iOS beta' release is marked as 'OVERBOOKED' and has a 'Dynamic release date' selected. The 'Start date' is set to 'As early as possible' and the 'Release date' is set to 'Dynamic release date'.

By deciding which method best applies to your process, you can plan upcoming goals and milestones with certainty that dates will be met.

If you are worried about several teams working on items for a shared release, then you can use a **cross-project release** option. This plan will include an individual release for each project, and keep them in sync. This reduces administration and makes for easier planning.



If you want to create a cross-project release, follow these steps:

1. Click **Create Release**
2. Select **Multi Project**
3. Select relevant projects for the release
4. Set **Start** and **End** dates

6. Keep data up to date

With your working plan you can manipulate data and instantly calculate the impact of change. Any changes made to issues in JIRA Software will automatically update the data in your portfolio plan. But think of Portfolio for JIRA as a sandbox where you can make changes and commit them back to JIRA Software. If you are not happy with those changes, you can always revert.



Review and commit changes

Change types: All ▾ Category: All ▾ 🔍

Changes 5 selected 1 4 0

Title	
<input checked="" type="checkbox"/> ADR-2 Establish basic app dev framework	Setup dev and and build environment UPDATED <input type="button" value="x"/>
<input checked="" type="checkbox"/> Android App Development Team	
<input checked="" type="checkbox"/> IOS-9 Trip management	
<input checked="" type="checkbox"/> IOS Visualise different scenarios	
<input checked="" type="checkbox"/> ADR-3 Setup dev and and build environment	

Date: 23/Nov/16 01:31 PM
Rank: Changed
Releases: Android 1.0
Android-beta
Team: IOS app Team

Whenever you make a change like move an epic into a different release, an orange indicator will flag the change and the number count of your uncommitted changes – seen in the upper left-hand corner of the plan – will increase. Click on the uncommitted changes to open the commit/revert dialog and once you are satisfied with all of your changes you can commit those changes into JIRA Software.

What-if scenario planning

The commit/revert feature opens the door for those that want to try out different scenarios and pick the best one. For example, if you want to find out how changes to release scope or added team members would impact your release dates you can sandbox the changes and calculate your plan under different variables.



Work with your plan

And that is it – you now have a plan! The schedule view (top half of the page) has been calculated on the scope, release, and team information supplied during the plan’s creation.

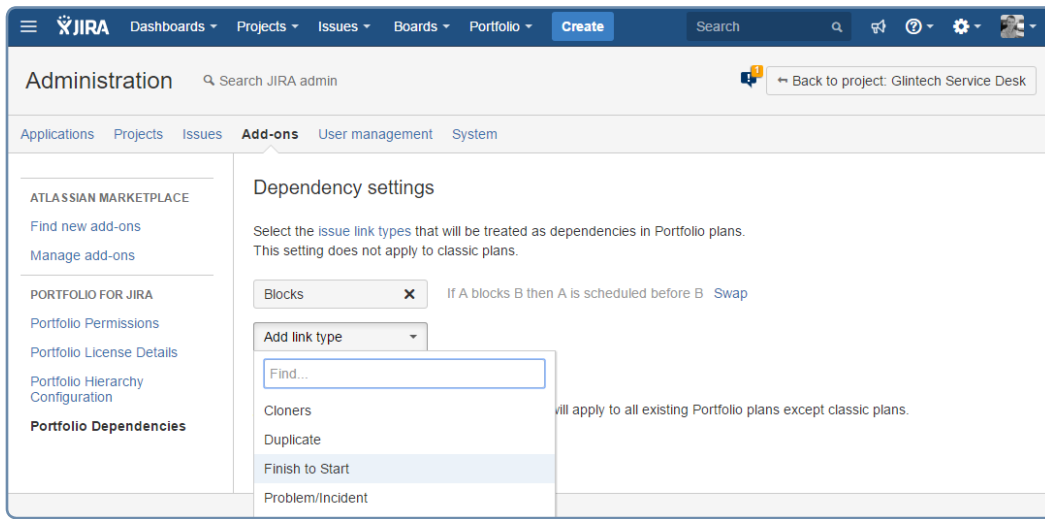
At this stage, you will have a basic plan where you can manipulate data and view a roadmap. But it does not stop here; there are more advanced features to accommodate many use cases. This section will highlight a handful of features you might find useful like the different filtering options on the schedule view. You can isolate certain releases, or split the view by team, to help draw attention where needed during planning.



Setting up dependencies

Dependencies can be created and visualised in Portfolio for JIRA using issue links. By default, only ‘Blocks’ and ‘Blocked by’ issue links control dependencies. You can add more **issue link types** for Portfolio for JIRA to consider dependencies through the Admin.

1. Navigate to **Administration > Add-ons > Portfolio Dependencies**
2. Add the **link type** you wish to include (check the order of blocking on the right)



Using plan configurations

To check out the scheduling options, go to **Plan > Configuration**. This is where you can tweak how Portfolio for JIRA assigns work. The main options to consider are:

- **Issue Assignee Import Level** – To set the JIRA ticket assignee to the team member assigned in your plan:
 1. Go to **Commit of changes** section under **Configuration**
 2. Enable **Commit issue assignee**
 3. Assign the ticket to your team member

The assignee of the ticket will only be set for issues assigned to them.

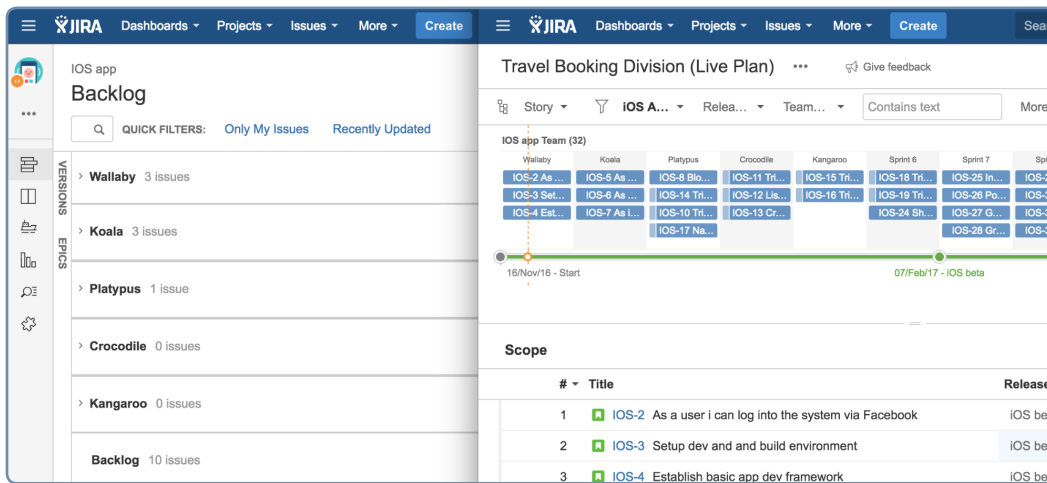
- **Dependent Story Constraint** – If issues have dependencies, Portfolio for JIRA will schedule them in different sprints (or on different days for Kanban) unless you say otherwise.
- **Enforce Concurrent Work** – If multiple team members are assigned to a single item, the scheduling algorithm ensures that all the work for that item is scheduled into a single sprint. Otherwise, individual team members' work can be scheduled whenever there is free capacity allowing a work item to span multiple sprints.



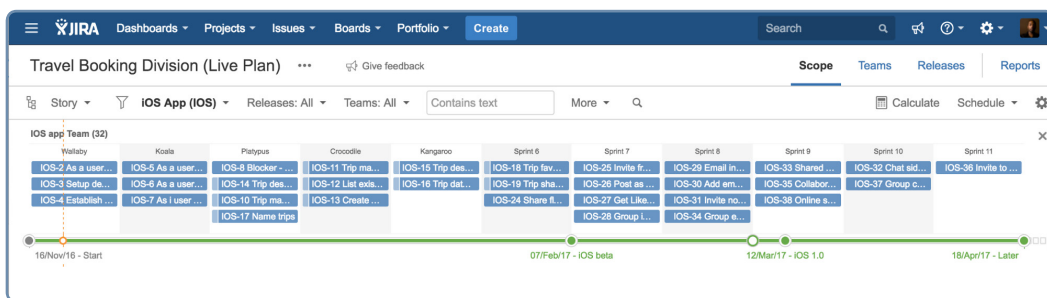
Planning with sprints

If you are working with sprints, you can use a board as an issue source. Sprints that you create in your board backlog are dynamically connected with Portfolio for JIRA and will be used to manage sprint assignments. You cannot create custom sprints from inside Portfolio for JIRA, so make sure you create your upcoming sprints from inside JIRA Software.

The rank of backlog sprints determines the order in which they will be scheduled. Sprint names, dates, and sprint assignment of issues are all surfaced in Portfolio for JIRA.



Note that, in Portfolio for JIRA, sprints start by default from the date you calculated the schedule. If your sprints start on a particular date then Portfolio for JIRA will schedule your sprint from your specified date and all future sprint cadences will be projected immediately following your active sprint.





From Portfolio for JIRA there are two ways to assign issues to a sprint depending on your usage.

- **Statically assign sprint** – The direct way to assign an issue to a sprint is using the sprint column. First, the team associated to the sprint must be assigned to the issue (either statically set or calculated). Then use the sprint column in the scope table to select the sprint.
- **Assign calculated sprint** – Alternatively, you can bulk-assign one or multiple issues calculated to a sprint. When you calculate a schedule and you want the forecasted sprints to be reflected on the JIRA board, simply click on the sprint name and select “assign X calculated issues and commit changes”.

The screenshot shows the Jira Portfolio interface for the 'Travel Booking Division (Live Plan)'. The main view is a capacity planning board for the 'iOS app Team (32)'. The team is divided into two members: 'Wallaby' and 'Koala'. Under 'Wallaby', there are three issues: 'IOS-2 As a user...', 'IOS-3 Setup de...', and 'IOS-4 Establish ...'. Under 'Koala', there are three issues: 'IOS-5 As a u...', 'IOS-6 As a u...', and 'IOS-7 As i us...'. A context menu is open over the 'Koala' member, displaying the following information:

- Koala**
- 30/Nov/16 - 13/Dec/16**
- Free capacity** 2 pts
- Planned capacity** 18 pts
- Capacity utilization** 90 %
- Assign 3 calculated issues** (with an information icon)

At the bottom of the board, a timeline shows the start of the sprint on 16/Nov/16.

Note: This will not work for issues that span multiple sprints or estimated epics that are exposed in the story-level view.



Plan with mixed estimates

If you have projects in your plan that follow different estimation methods there is a functionality to accommodate such a use case. Go to **Plan > Configuration > Issue sources**. You can set a conversion ratio for each issue source that uses a different estimation method from your plan. But how do you set a ratio? For a mixed estimate plan we recommend you keep your plan in time-based estimation and follow these steps to set your story-point to hours conversion ratio:

The screenshot shows the Jira interface for configuring a plan. The main content area is titled "Configure Travel Booking Division (Live Plan)". On the left, there is a navigation menu with sections: PLANNING (Stages and skills, Scheduling, Working hours and days), SOURCE DATA (Issue sources, Custom fields, Commit of changes), and ACCESS CONTROL (Permissions). The "Issue sources" section is active, displaying a list of six issue sources. Each source has a "Board" dropdown, a text input field for the source name, and a link for "Advanced settings".

Board	Issue Source	Conversion Ratio
Board	Platform Development	Advanced settings
Board	IOS app	Convert 1 story point into 6.67 hours
Board	Android App Development	Advanced settings
Board	Performance Engineering	Convert 1 story point into 7 hours
Board	Portfolio Management	Advanced settings
Board	Team Travel Web	Advanced settings

An "Edit sources" button is located at the bottom of the list.

Story points to hours

1. Determine the total weekly capacity of your team (e.g five team members with 40-hour capacity each per week = 200hr team capacity).
2. Divide the team's sprint velocity for 1 week by the weekly capacity (e.g one-week velocity = 30 so $200/30 = 6.667$).
3. Use this result as your conversion ratio.

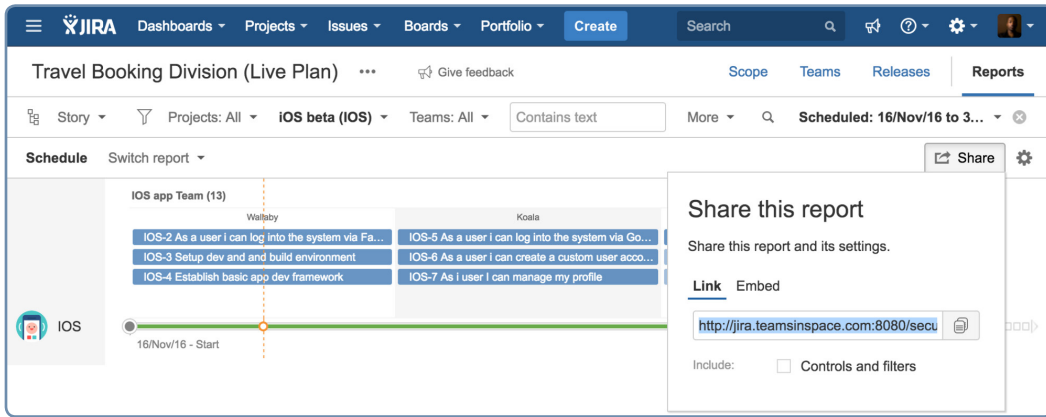
This is not necessarily your final conversion ratio. As your team works over time, keep this ratio up to date to better represent your team performance. You can also convert hours to story points, but that may take some trial and error to determine the right velocity (and therefore conversion ratio) to use.



Reporting on your plan

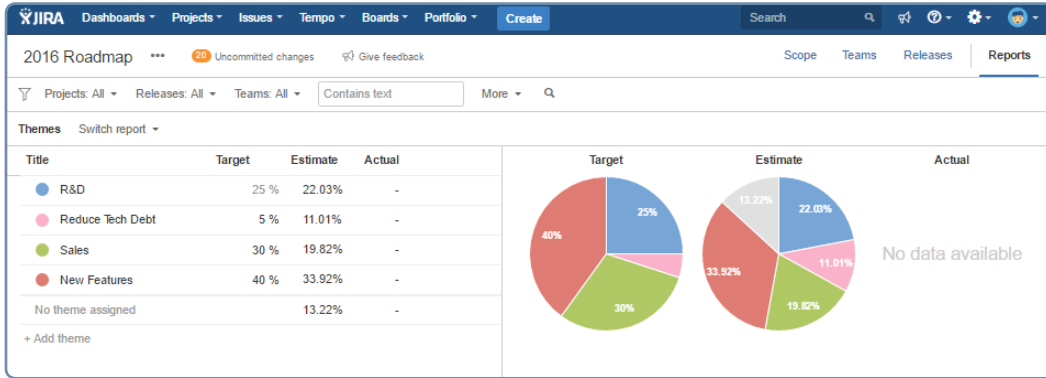
Click on the reporting tab to visualize and share your data with six different reports: themes, releases, schedule, capacity, sprints, and scope.

Reports are useful to dig deeper into your data through these dedicated views and share a persistent view of this data. For example, say you want to share the roadmap for just the iOS team with your manager for a specific date range. You can filter your plan to just see the relevant data, then generate a URL or embeddable iFrame to share.

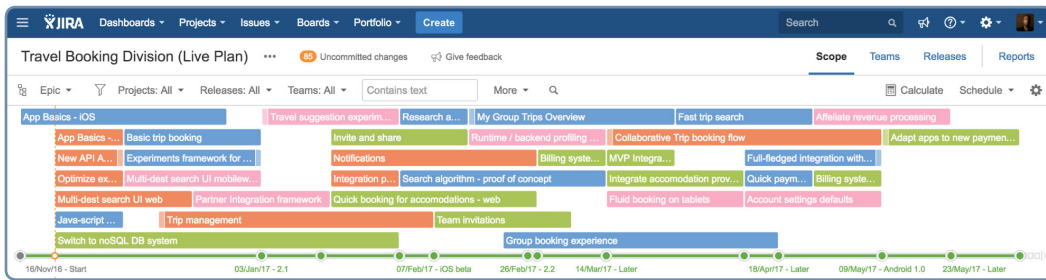




If you need to report on how a your roadmap items relate back to an organization-wide focus area, then you might want to use **Themes**. These track how your work measures against your strategic objectives. Get real-time comparison between your target distribution of work against actual work that gets completed.



One advantage of using themes is that you can visualize your schedule with each item color-coded by theme:





A non-perfect world

While all of this is great in theory, many teams have dependencies and factors that affect their roadmap that are external from JIRA Software. It may be other planning tools, third-party vendors, or simply a part of the business that is not ready to change.

If you have a dependency or external milestone that you need to reflect in your plan, try using the earliest start date function. This will pin that task into the timeline, and Portfolio for JIRA's scheduling algorithm will work around it.

The screenshot shows the JIRA Portfolio interface for the 'FY17 Plan'. The main view is a roadmap with a timeline from 12Jul16 to 10Jan17. Key tasks and their dependencies are visible:

- ADR-15 My Group Trips Overview (Starts 20/Oct/16)
- IOS-9 Trip management (Starts 20/Oct/16)
- IOS-20 Group booking experience (Starts 20/Oct/16)
- ADR-8 Invite and share (Starts 30/Nov/16)
- IOS-21 Team invitations (Starts 10/Jan/17)
- IOS-22 Social media integrations (Starts 10/Jan/17)

The 'Trip management' task details are shown in the right-hand pane:

- Title:** Trip management
- Parent:** Initiative: IOS-23 New mobile pl...
- Description:** Click to add description
- Dependencies:** No dependencies
- Scheduling:** Earliest start: No constraint; Start date: 20/Oct/16; End date: 30/Nov/16



How does Portfolio for JIRA calculate your schedule?

Our scheduling algorithm takes the data you provide to produce a realistic, data-driven roadmap. While a roadmap can simply be produced by only using estimates and team velocity, we let you get much more granular with the variables you provide.

So that you do not get lost in the details, here is a list of all the variables our algorithm considers and what they mean. Do not stress if you do not use all or even most of them. The level of detail you provide Portfolio for JIRA depends on your team.

- **Estimate:** The expected work effort measured in story points, hours, or days.
- **Team velocity:** How much work effort or how many hours a team can complete in a regular cadence.
- **Priority:** The rank of items in the scope table reflect the issue's priority. Drag and drop items to re-prioritize them.
- **Release:** Assignment of issues to releases and the start and end dates of releases.
- **Skills:** Amount of a work item's estimate that requires a particular skill. Skills can be associated with individual team members.
- **Work stages:** Activities that can happen either in parallel or sequential activities.
- **Availability:** Teams and team member availability.
- **Dependencies:** Blocked or blocking items between backlog items.
- **Estimate conversion:** A per-issue-source conversion rate when planning with mixed estimates.
- **Configurable constraints:** For example, how many people can work in parallel on a story.



Take-aways

There is a lot to consider outside of Portfolio for JIRA and when using the tool, but creating a long-term plan is essential to successful agile software development. Just remember these points to planning:

- **Decide what you want** – Spend time on how to configure your plan initially, and you will reap the rewards later.
- **Plan and re-plan** – Things change, your plan will too.
- **Don't worry too much about the distant future** – start with next week and go from there.
- **Communicate** – This plan affects everyone. Share the love!

Above all else, be agile: Start planning, look at how you did and improve. Remember, planning is an art, not a science. Practice make perfect.

Sources and resources: You can find more information on the topics covered in this paper in our Atlassian Documentation (www.confluence.atlassian.com).

