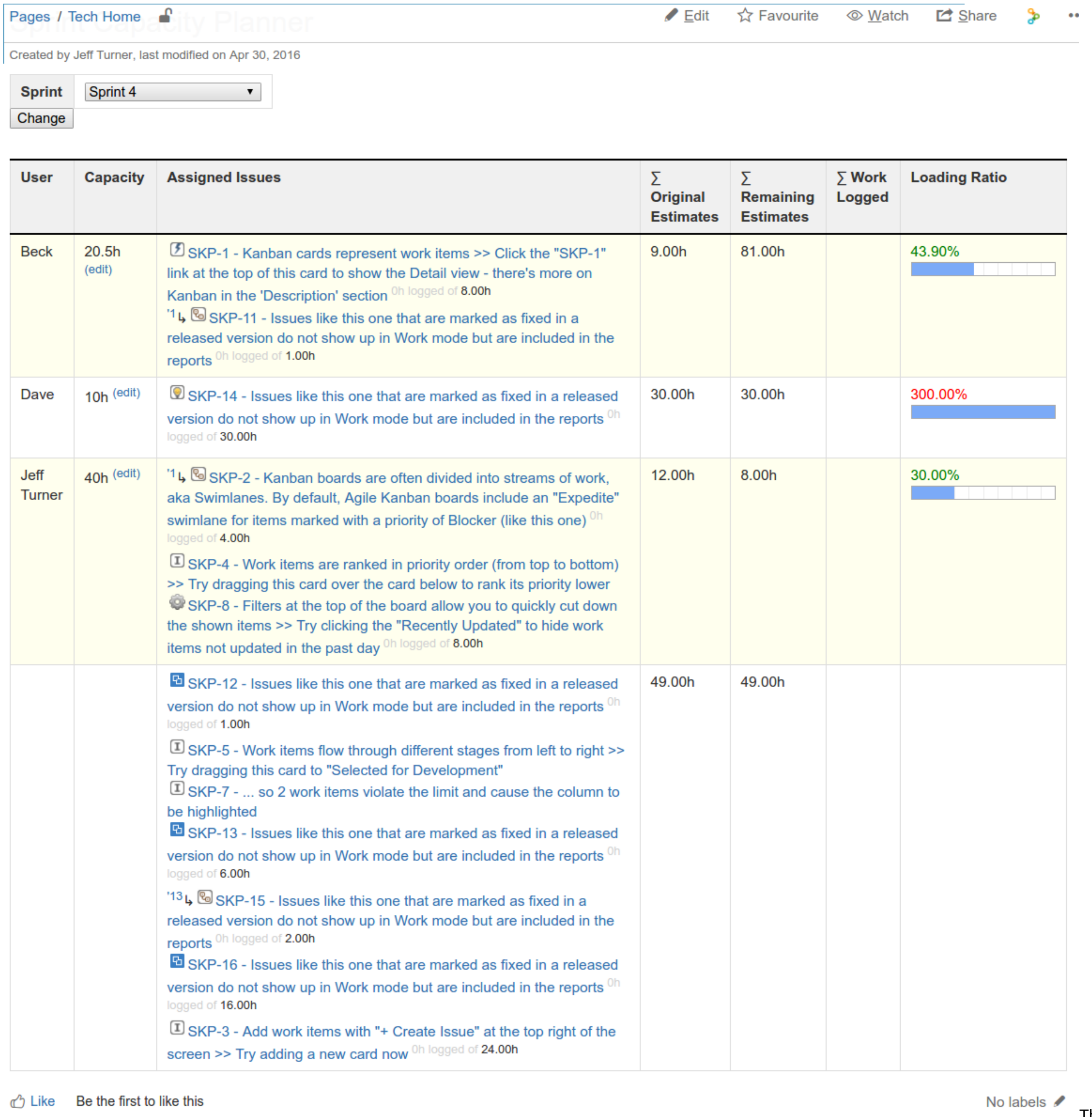
JIRA Sprint Capacity Planner

Planning a sprint in JIRA Agile involves assigning issues. You want to spread the load fairly, not overburdening or underburdening workers. To complicate matters, some issues take longer than others (with more Story Points or greater Time Estimate), and the capacity of each worker varies per week, as they may have time off or other responsibilities.

To meet this planning need in a simple way, we have developed the Sprint Capacity Report, a Confluence-based report that queries JIRA, showing issues grouped per assignee, showing time estimates vs. assignee capacity.The full implementation is described below. The only requirement is that you have the SQL for Confluence plugin, which is worth buying anyway, and that your database be PostgreSQL. For production use, Red Radish Consulting offers installation, customization and support services.



Created by Jeff Turner, last modified on Apr 30, 2016

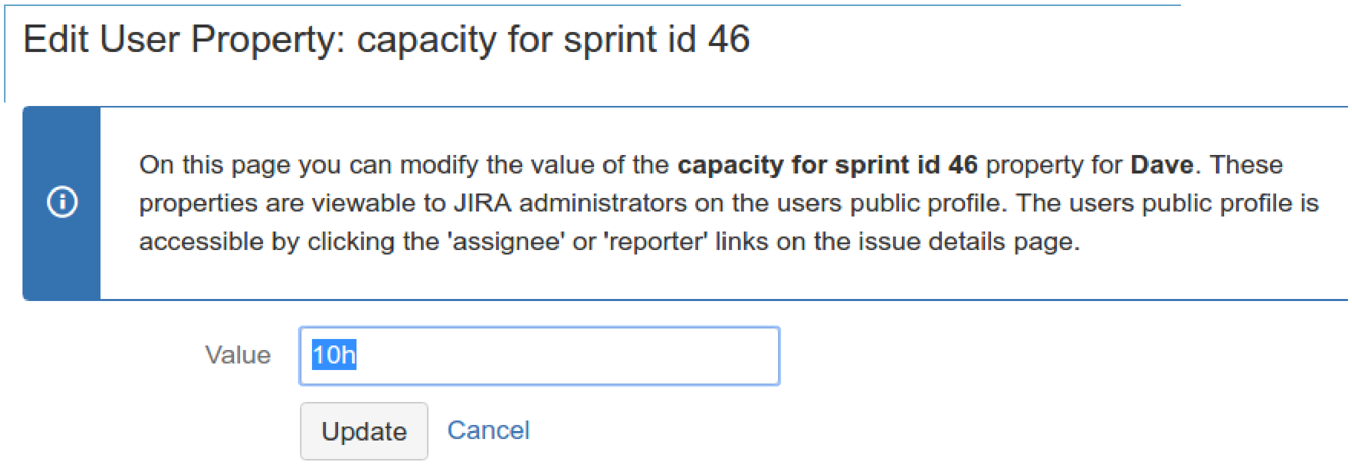
Sprint: Sprint 4

Change

User	Capacity	Assigned Issues	∑ Original Estimates	∑ Remaining Estimates	∑ Work Logged	Loading Ratio
Beck	20.5h (edit)	SKP-1 - Kanban cards represent work items >> Click the "SKP-1" link at the top of this card to show the Detail view - there's more on Kanban in the 'Description' section 0h logged of 8.00h '1↳ SKP-11 - Issues like this one that are marked as fixed in a released version do not show up in Work mode but are included in the reports 0h logged of 1.00h	9.00h	81.00h		43.90%
Dave	10h (edit)	SKP-14 - Issues like this one that are marked as fixed in a released version do not show up in Work mode but are included in the reports 0h logged of 30.00h	30.00h	30.00h		300.00%
Jeff Turner	40h (edit)	'1↳ SKP-2 - Kanban boards are often divided into streams of work, aka Swimlanes. By default, Agile Kanban boards include an "Expedite" swimlane for items marked with a priority of Blocker (like this one) 0h logged of 4.00h SKP-4 - Work items are ranked in priority order (from top to bottom) >> Try dragging this card over the card below to rank its priority lower SKP-8 - Filters at the top of the board allow you to quickly cut down the shown items >> Try clicking the "Recently Updated" to hide work items not updated in the past day 0h logged of 8.00h	12.00h	8.00h		30.00%
		SKP-12 - Issues like this one that are marked as fixed in a released version do not show up in Work mode but are included in the reports 0h logged of 1.00h SKP-5 - Work items flow through different stages from left to right >> Try dragging this card to "Selected for Development" SKP-7 - ... so 2 work items violate the limit and cause the column to be highlighted SKP-13 - Issues like this one that are marked as fixed in a released version do not show up in Work mode but are included in the reports 0h logged of 6.00h '13↳ SKP-15 - Issues like this one that are marked as fixed in a released version do not show up in Work mode but are included in the reports 0h logged of 2.00h SKP-16 - Issues like this one that are marked as fixed in a released version do not show up in Work mode but are included in the reports 0h logged of 16.00h SKP-3 - Add work items with "+ Create Issue" at the top right of the screen >> Try adding a new card now 0h logged of 24.00h	49.00h	49.00h		

The drop-down list selects the Sprint whose issues/assignees are shown. In this example we see, for instance, that Dave is able to work 10 hours this sprint, is assigned one issue (SKP-14), which is estimated to take 30h, so Dave would be at 300% capacity unless we change things. There are 49 hours' worth of unassigned issues.

User capacities are stored as user properties in JIRA. E.g. clicking 'Edit' in the Capacity column for Dave brings up:



Edit User Property: capacity for sprint id 46

On this page you can modify the value of the **capacity for sprint id 46** property for **Dave**. These properties are viewable to JIRA administrators on the users public profile. The users public profile is accessible by clicking the 'assignee' or 'reporter' links on the issue details page.

Value: 10h

Implementation

The report is implemented as a giant parametrized SQL query running within a Confluence page, with the SQL running against the JIRA database. The only dependency is the commercial [SQL for Confluence](#) plugin. Advanced users are encouraged to follow and understand the implementation described below. Advanced users *who know the value of their own time* are encouraged to [contact

Install the SQL plugin

First install the SQL for Confluence plugin. This plugin is indispensable for Confluence, much like ScriptRunner is for JIRA.

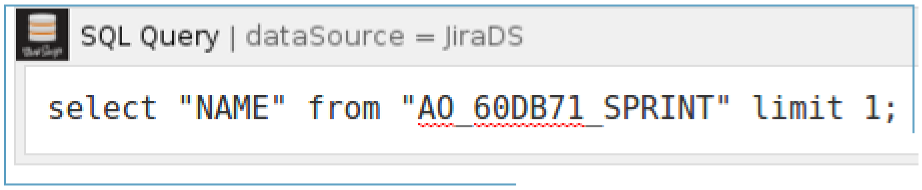
Define a Datasource to access JIRA

First we need to allow Confluence access to our JIRA database, via a datasource configured in Confluence's `server.xml` file, usually found at `/opt/atlassian/confluence/conf/server.xml`. This block should be inserted nested within the `<Context>`, just below the line `<Manager pathname="" />`.

conf/server.xml

```
                    <Resource name="jdbc/JiraDS" auth="Container" type="javax.sql.DataSource"
                           username="redradish_jira"
                           password="s3cret"
                           driverClassName="org.postgresql.Driver"
                           url="jdbc:postgresql://localhost/redradish_jira"
                           maxActive="20"
                           validationQuery="select 1"/>
```

Restart Confluence. Afterwards, a simple query like `SELECT "NAME" FROM "AO_60DB71_SPRINT"` against the `JiraDS` database should work.



SQL Query | dataSource = JiraDS

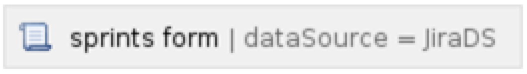
```
select "NAME" from "AO_60DB71_SPRINT" limit 1;
```

Sprints drop-down

We need to prompt the user for the Sprint they wish to review.

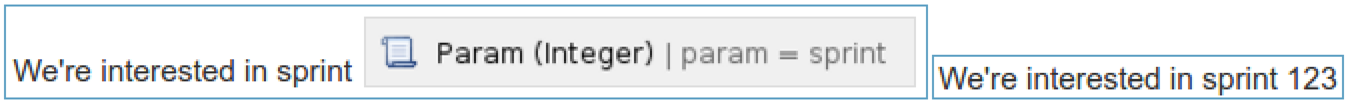
This drop-down is implemented as a Confluence "user macro". See the JIRA Sprint select-list macro for the implementation.

Once implemented, insert the macro in your page, parametrized with the JIRA datasource name (`JiraDS`):


sprints form | dataSource = JiraDS

Define a Param user macro

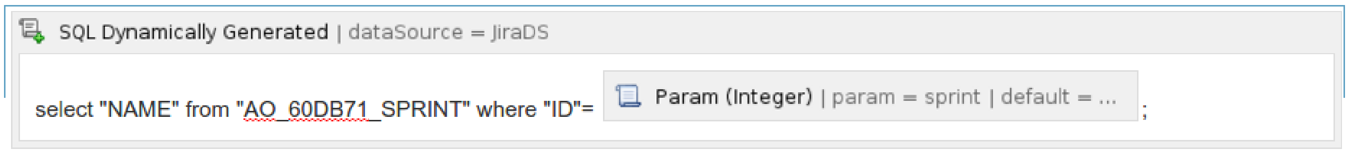
When the user clicks a sprint from the list above, the page reloads with a `sprint=xyz` parameter, where `xyz` is the internal Sprint ID. Define the **Param (Integer)** macro, as specified at HTTP Parameter Macros, and verify that it works.


We're interested in sprint Param (Integer) | param = sprint

We're interested in sprint 123

Make a dynamic SQL wrapper macro

We now know what sprint ID we're interested in, and could potentially query the right database table with the SQL macro, but unfortunately the SQL Query macro won't allow nesting of our `Param (Integer)` macro. See Allow macro content inside any other macro for a SQL wrapper macro that gets around this. We can now do:


SQL Dynamically Generated | dataSource = JiraDS

select "NAME" from "AO_60DB71_SPRINT" where "ID"= Param (Integer) | param = sprint | default = ... ;

Query SQL

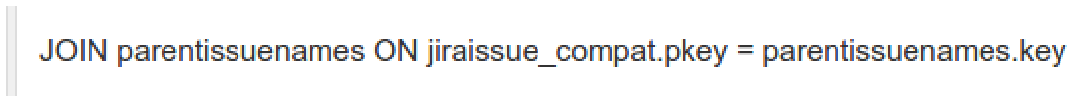
Create a page in Confluence called **Sprint Capacity Planner**, whose contents is a **Sprints Dropdown** macro, and then a **SQL Dynamically Generated** macro wrapping a **Param (Integer)**:

Sprint Capacity Planner

Sprints Dropdown

SQL Dynamically Generated | dataSource = JiraDS | output = xhtml | printsql = false

```
WITH

sprintparam AS (

select   Param (Integer) | param = run_dynamic_sprint ...   ::integer AS id),

defaultsprint AS (

select "ID" AS id from "AO_60DB71_SPRINT" ORDER BY "ID" DESC LIMIT 1

),

sprintid AS (SELECT COALESCE(sprintparam.id, defaultsprint.id) AS id from sprintparam CROS

props AS (select propertyvalue AS baseurl from propertyentry pe JOIN propertystring ps ON pe.i

jiraissue_compat AS (SELECT (project.pkey::text || '-'::text) || ji.issuenum AS pkey, ji.id, ji.issuety
ji JOIN project ON ji.project=project.id),

parentissuenames AS (

select encodehtml(ji.pkey) as key

from jiraissue_compat ji

JOIN (select * from issuetype  where coalesce(pstyle, '')!='jira_subtask') AS issuetype ON ji.issue

join customfieldvalue cfv on cfv.issue = ji.id and cfv.customfield = (select id from customfield whe

join sprintid on cfv.stringvalue = cast(sprintid.id as varchar)

),

parentissues AS (

SELECT

null::varchar AS parentkey,

null::varchar AS parentsummary,

jiraissue_compat.*

FROM jiraissue_compat
```



```
JOIN parentissuenames ON jiraissue_compat.pkey = parentissuenames.key
```

To create this page, install the Confluence Source Editor plugin, save and re-edit your **Sprint Capacity Planner** page, click the <> in the top-right, and paste in the contents of sprintcapacityplanner.xhtml

Conclusion

The end result is a Confluence page where:

1. A drop-down list of Sprints is auto-generated by querying the JIRA database.
2. When the user clicks 'Change', the chosen sprint's ID is re-sent to the page as a URL parameter, which is extracted by the Param macro.
3. The sprint param is embedded in a large SQL query, which queries the JIRA database to extract relevant sprint data

Please contact us if you have any comments or questions.