

Yoda

Agile Project Management with GitHub

Jens Vedel Markussen, Engineering Manager
Hewlett Packard Enterprise

Introduction

Yoda was developed during 2017 at **Hewlett Packard Enterprise** to support **Agile Project planning** and execution for development of a new innovative product.

GitHub was already in place for **source code versioning** and **issue tracking** (bugs and features).

The ambition was to **enhance GitHub** to become an **all-in-one solution** for Agile Project Planning and Execution.

Yoda **augments GitHub** by adding **estimates** and **sprint planning** to issues. Further, Yoda brings various tools for **issue reporting** and management.

Yoda was **Open-Sourced** using an MIT license in January 2018.

Content

- Agile Project Management
- Stories, Features, Epics, ... in GitHub (issues)
- Sprints in GitHub (milestones)
- Story point estimation in Github Issues
- GitHub issue labelling convention
- Yoda Reporting Tools
 - Issue Time Statistics, CFD, Issue Exporter
- Yoda Agile Project Management Tools
 - Burndown Chart, Velocity Chart, Kanban Board
- Other Yoda tools
 - Label Manager, Admin, Task Copy
- Yoda Architecture

Agile Project Management

- **Agile** project management is becoming an industry **de-facto standard**
- Project- and product-**development** happens as a series of **sprints**.
- Software is **released** either at the end of each sprint, or every *n'th* sprints as a **product increment**.
- Sprints address (user) **stories**, which are estimates using **story points**.
- Often **SCRUM** methodology drives development.
- Different frameworks, e.g. **SAFe** (Scaled Agile Framework) add descriptions at higher level than (user) stories to capture required functionality (**Epics, Capabilities, Features**).

(User) Stories, etc. in GitHub

- GitHub **issues** can be used to represent (User) **Stories** – and as well Epics, Capabilities, and Features
- GitHub Issues bring many **relevant features** for this, e.g.
 - Web UI, Markdown, graphics, discussions, assignments, labels, lists, file attachments, references, milestones, etc....
- GitHub **issue references** can be used to link descriptions (e.g. stories x and y required to implement Epic z gives references $x \leftrightarrow z$ and $y \leftrightarrow z$).

Sprints in Github

- A **sprint** defines a time period (typically 2-4 weeks) in which a number of user stories (as broken down into tasks) are delivered.
- **Yoda** uses Github **milestones** for sprints
 - Milestones already have an end/due date.
 - Yoda expects to have as well a sprint **start date**
 - Optionally, a team sprint capacity figure (in story points)

github-yoda / demo

<> Code Issues 29 Pull requests

Labels Milestones

Title

Sprint 2

Due date (optional)

16-03-2018

Description

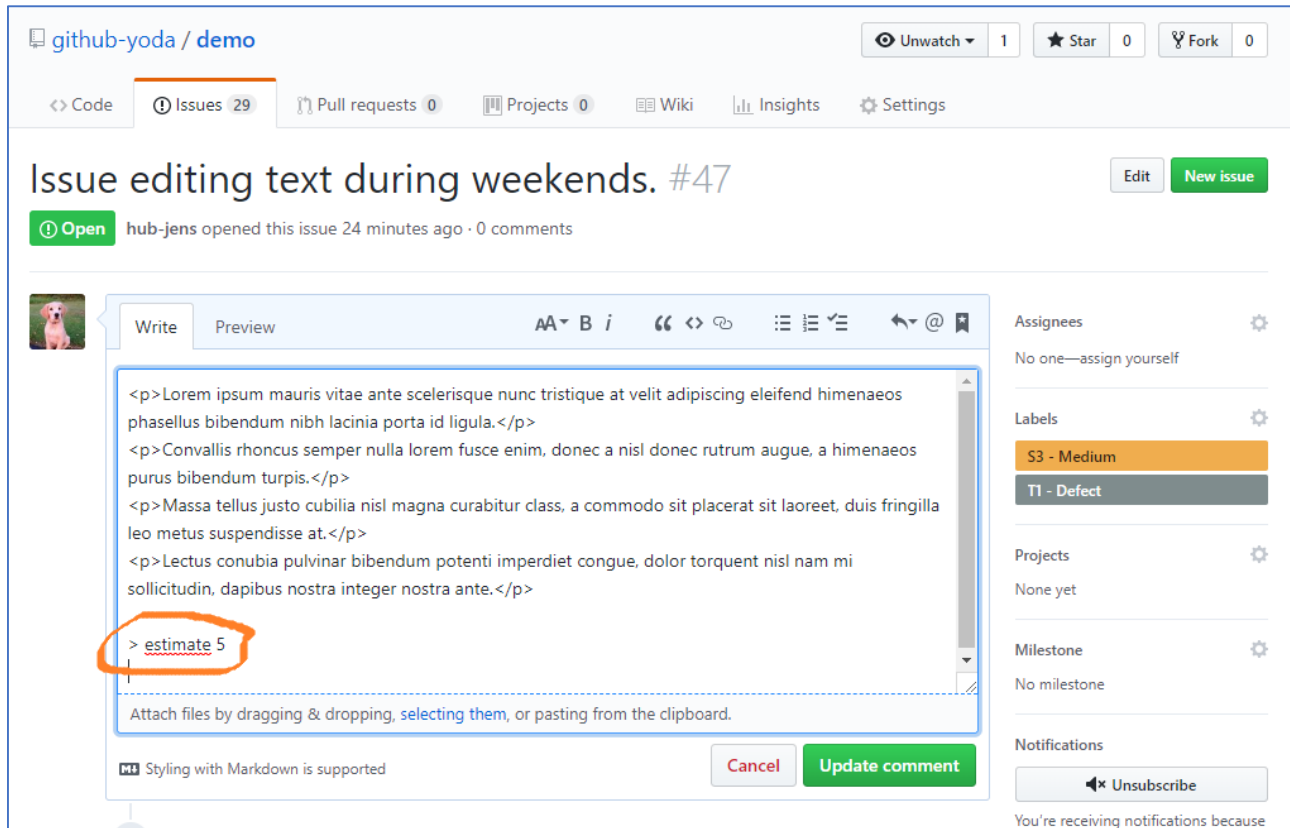
```
> startdate 2018-03-01
> capacity 65
|
```

Story point estimation in Github Issues

- Github issues for (User) Stories **do not have** a dedicated **field** to store estimates (story points).
 - Similarly, no features exists for **summing up estimates** (into milestones, projects, etc.)
- Instead Yoda introduces **two options** for handling estimates into issues:
 1. As **special text** "> estimate (story points)", in the body (first comment) of the issue
 2. Using one of several fixed **story point labels**.
- If using labels, suggest to create labels with **Fibonacci-like** values (1,2,3,5,8,13,20,40) as typically done for Story Points.
- Yoda considers as well the **remaining effort** for an issue. If not provided the **remaining effort** is assumed to be **equal** to the **estimate** while the issue is **open** and **zero** when it is **closed**.
 - If using option 1 above (estimate into issue body), it is possible to specify as well one or more explicit remaining values using a "> remaining YYYY-MM-DD (story point value)" syntax.

Estimate example (body text)

Markdown "> estimate (story point)" format)

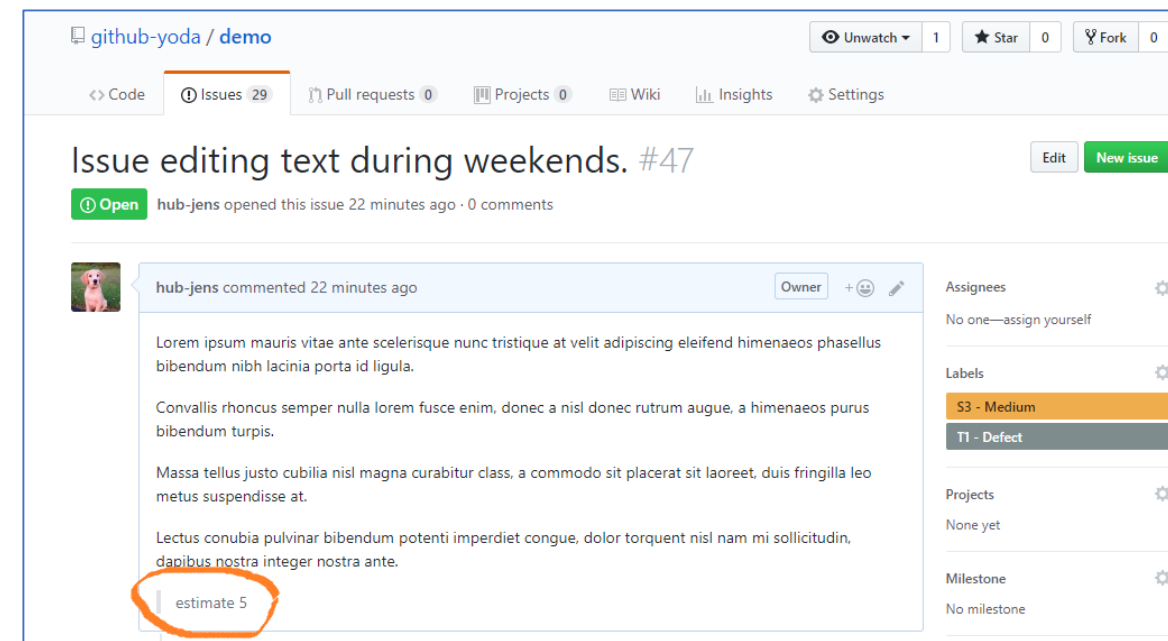


The screenshot shows the GitHub issue editor for the repository 'github-yoda / demo'. The issue title is 'Issue editing text during weekends. #47'. The issue is open and was created 24 minutes ago. The editor is in 'Write' mode, showing a rich text editor with a toolbar. The text input field contains the following Markdown code:

```
<p>Lorem ipsum mauris vitae ante scelerisque nunc tristique at velit adipiscing eleifend himenaeos phasellus bibendum nibh lacinia porta id ligula.</p>  
<p>Convallis rhoncus semper nulla lorem fusce enim, donec a nisl donec rutrum augue, a himenaeos purus bibendum turpis.</p>  
<p>Massa tellus justo cubilia nisl magna curabitur class, a commodo sit placerat sit laoreet, duis fringilla leo metus suspendisse at.</p>  
<p>Lectus conubia pulvinar bibendum potenti imperdiet congue, dolor torquent nisl nam mi sollicitudin, dapibus nostra integer nostra ante.</p>
```

The text '> estimate 5' is circled in orange in the input field. Below the input field, there is a note: 'Attach files by dragging & dropping, selecting them, or pasting from the clipboard.' The editor also has 'Cancel' and 'Update comment' buttons.

Resulting preview/HTML



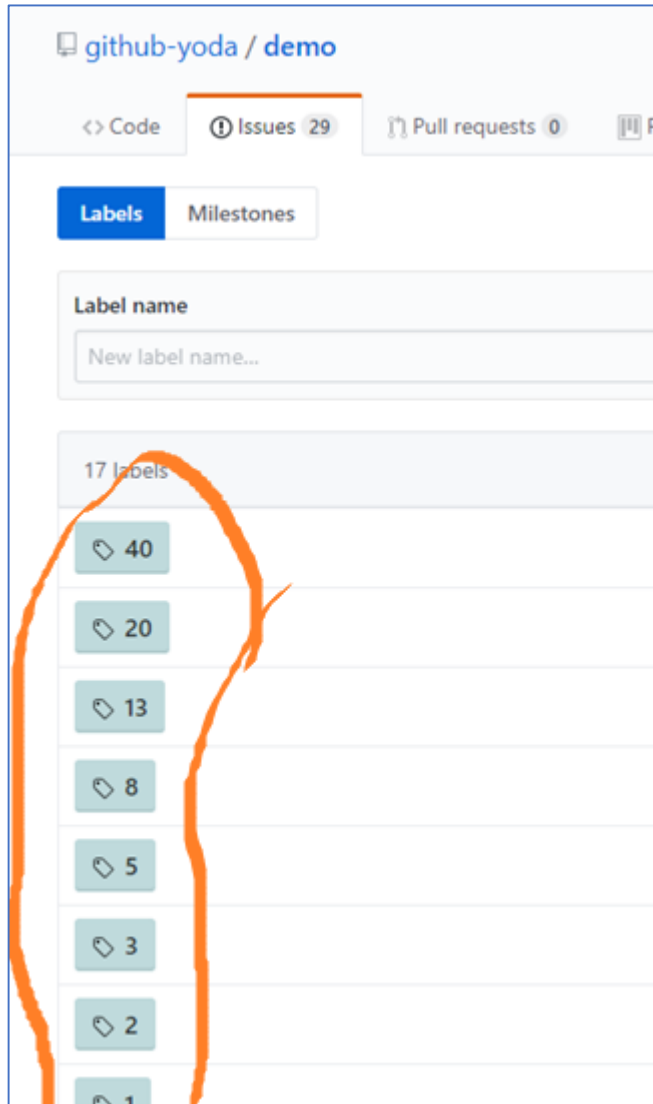
The screenshot shows the GitHub issue preview for the repository 'github-yoda / demo'. The issue title is 'Issue editing text during weekends. #47'. The issue is open and was created 22 minutes ago. The preview shows the rendered HTML for the text that was circled in the previous screenshot:

```
> estimate 5
```

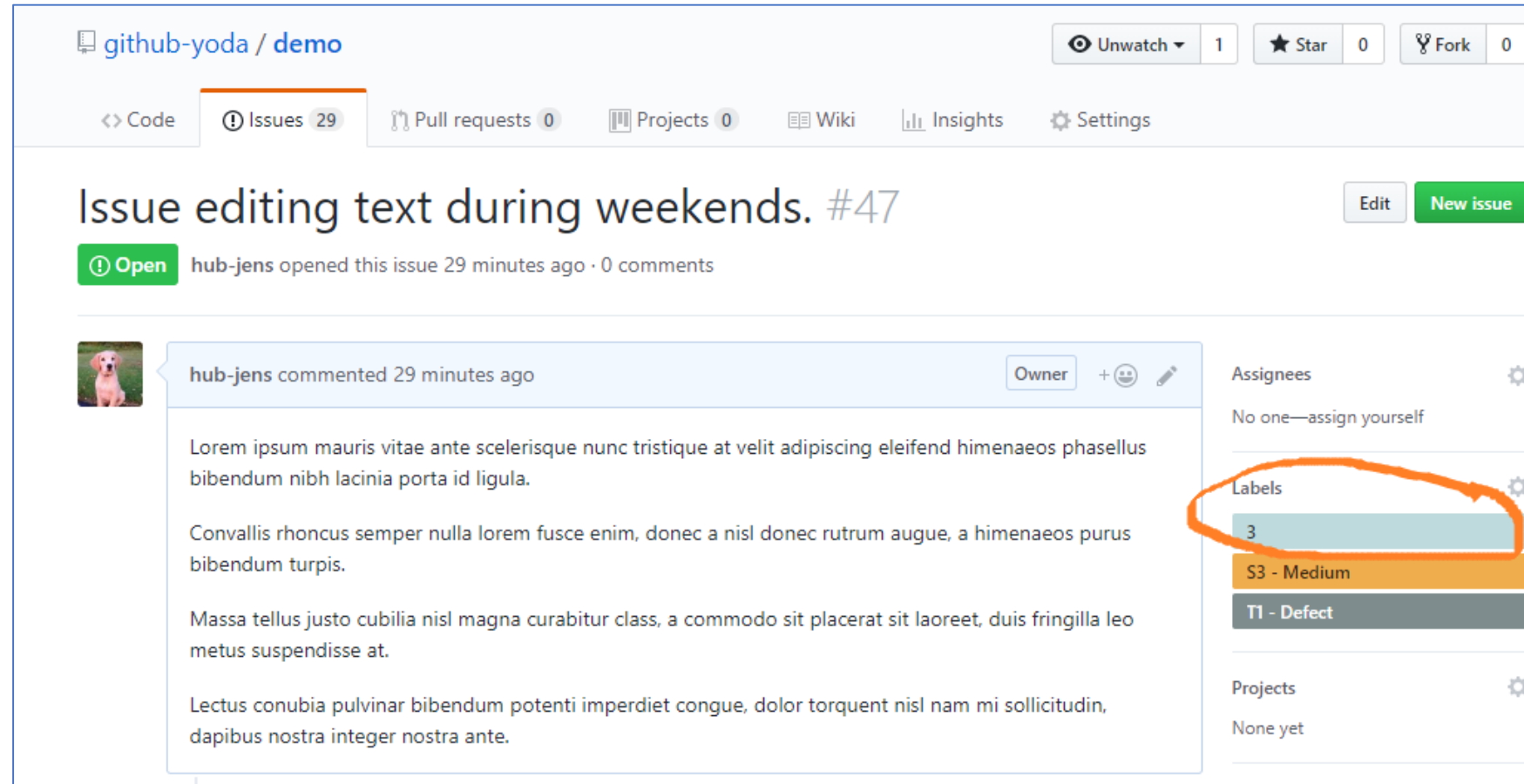
The text is rendered as a single line of text. The preview also shows the issue's metadata, including the issue number, the user who created it, and the issue's status. The preview also shows the issue's labels, which are 'S3 - Medium' and 'T1 - Defect'. The preview also shows the issue's projects, which are 'None yet'. The preview also shows the issue's milestone, which is 'No milestone'. The preview also shows the issue's notifications, which are 'Unsubscribe'.

Estimate example (using labels)

Fibonacci Labels



Issue with label estimate



Remaining example

Markdown "> remaining YYYY-MM-DD (story point)" format)

The screenshot shows the GitHub issue editor for the issue "Enhance topic list on test systems. #32". The editor has a "Write" tab and a "Preview" tab. The "Write" tab is active, showing a text area with placeholder text and a list of items. The list items are:

- > estimate 30
- > remaining 2017-12-15 22
- > remaining 2017-12-22 14
- > remaining 2018-01-08 4

The list items are circled in orange. Below the text area, there is a message: "Attach files by dragging & dropping, selecting them, or pasting from the clipboard." At the bottom, there are buttons for "Cancel" and "Update comment".

Resulting preview/HTML

The screenshot shows the GitHub issue preview for the issue "Enhance topic list on test systems. #32". The issue is marked as "Open" and was opened by "hub-jens" 2 days ago. The preview shows the rendered HTML of the issue content. The list items are:

- > estimate 30
- > remaining 2017-12-15 22
- > remaining 2017-12-22 14
- > remaining 2018-01-08 4

The list items are circled in orange. The preview also shows the rendered HTML of the placeholder text.

GitHub issue labelling convention

- To get maximum benefit from Yoda, it is important to be **consistent** on the **use of labels**. This is best done by having a **labelling** convention.
- Suggestion for a labelling convention is to assign to issues:
 - A **type label** (e.g. Defect, Enhancement, Tasks).
 - A **severity label** (e.g. Urgent, High, Medium, Low).
 - **Note:** These labels are mutually exclusive by convention not enforced by Github.
 - Optionally, use a prefix (e.g. T or S) for different label enumerations.

Example



Yoda Reporting Tools

Issue Time Statistics, CFD, Issue Exporter

Issue Time Statistics

- This report shows open GitHub **issues over time** in a **bar-chart**
- Issues can be **split** into different **bars** based on **labels** (e.g. Severity)
- Issue **label filters** can be applied
- **Start-** and **end-dates**, reporting **interval**, etc. can be adjusted
- Optionally, number of **opened** or **closed** reports during an interval can be **reported** instead of # of open issues.

Example: Issue Time Statistics

Owner

Repo

GitHub user

GitHub token

Label filter

Count

jens-markussen

.....

T1 - Defect

Issues

Days open

Opened

Closed

Start date (blank=2m ago)

End date (today=blank)

Interval

Label Bar Splitting

Other (blank to omit)

Title

Stacked

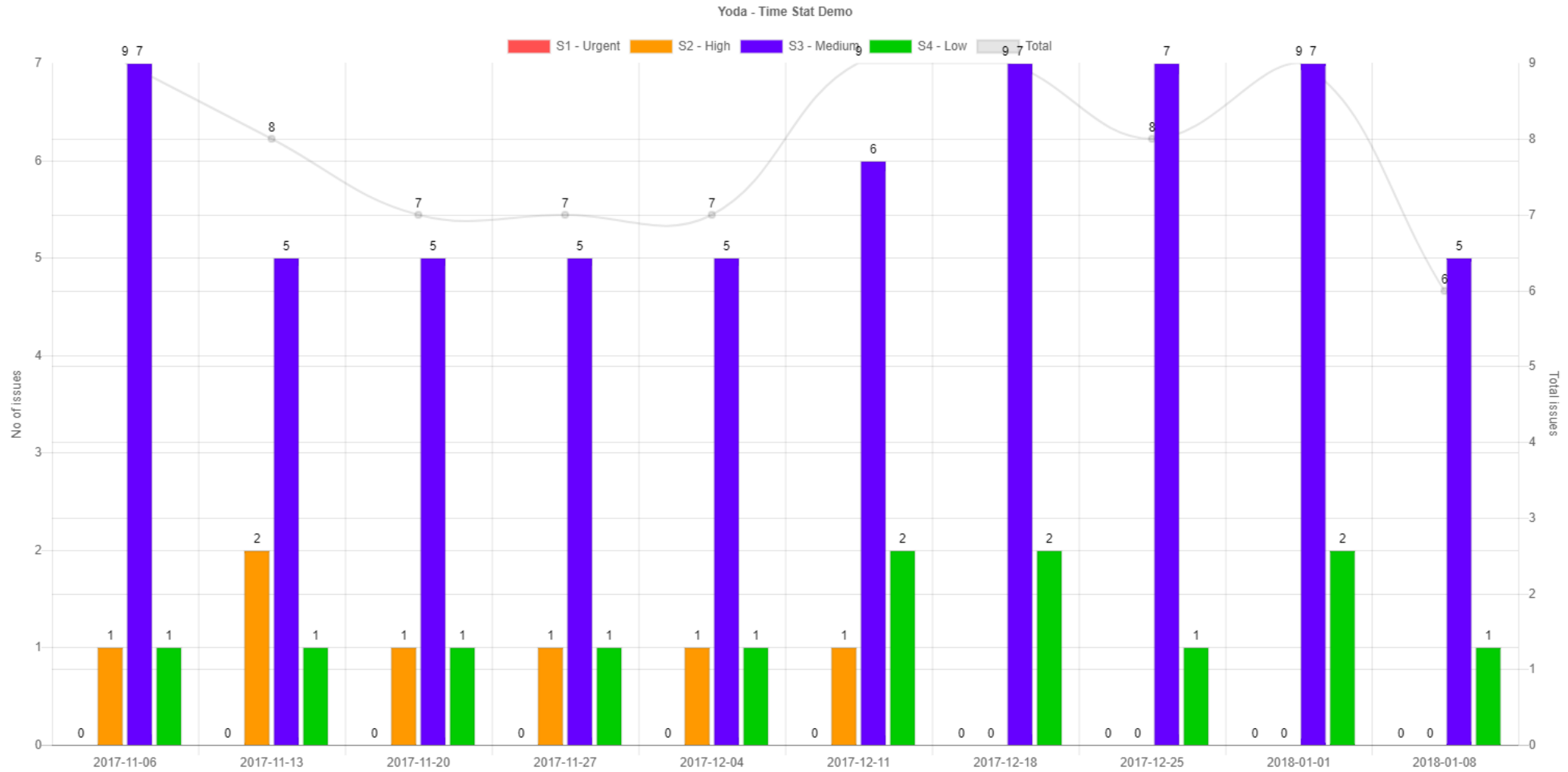
Draw chart

7

^S[1-4] -

Yoda - Time Stat Demo

☐



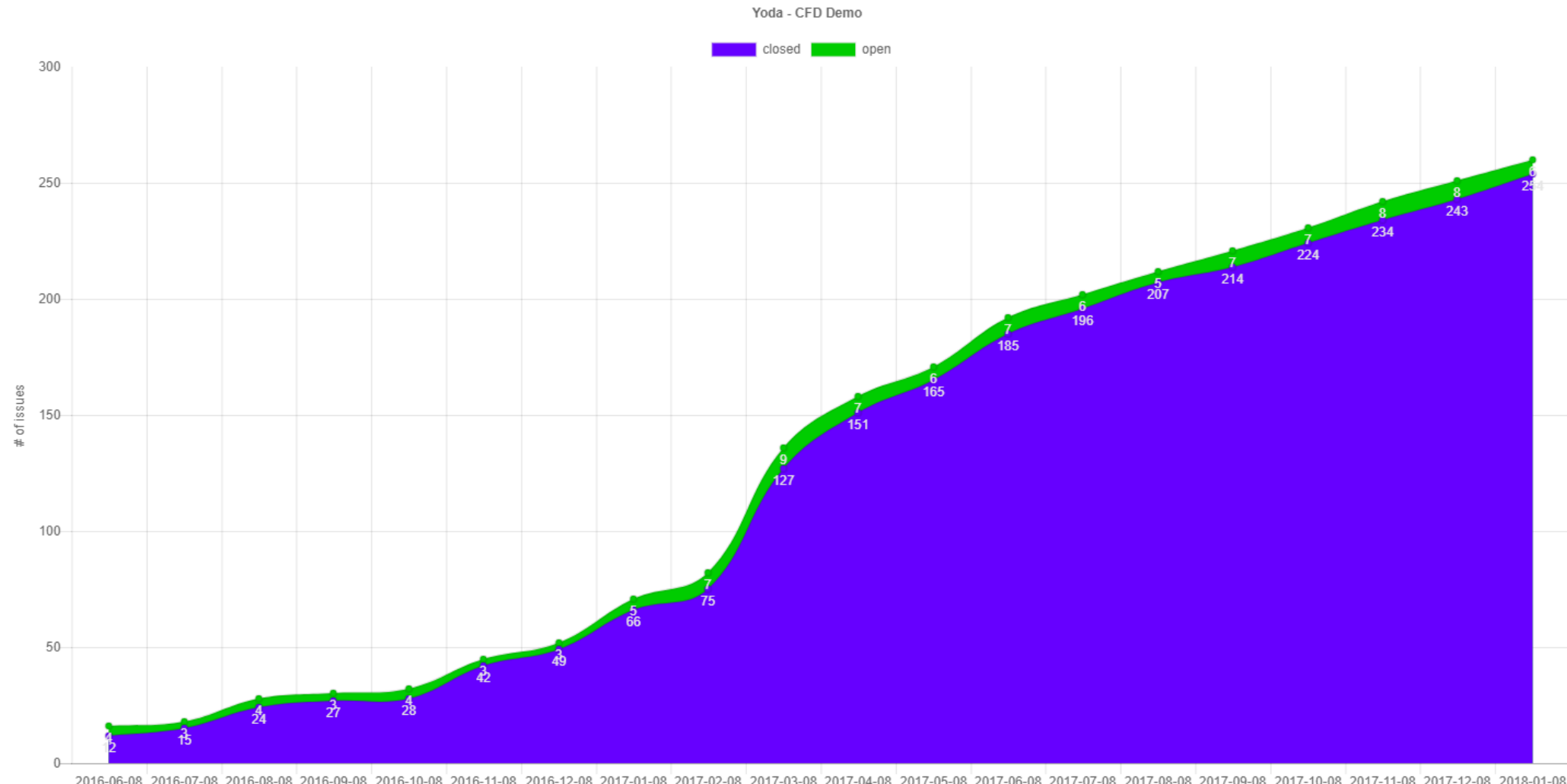
CFD (Cumulative Flow Diagram)

- A **CFD** shows **cumulative** number of **issues over time** split by state (open/closed)
 - **Normally** CFD charts may consider **more than just two states** (e.g. Open, In design, in development, in test, done/closed).
 - As GitHub only has two issue states (open and closed). **Yoda CFD only** uses these **two states**.
- Yoda can also draw the related **lead-time graph**.
 - This shows the **average** number of **days** an issue remained in the **open state**.

CFD Example

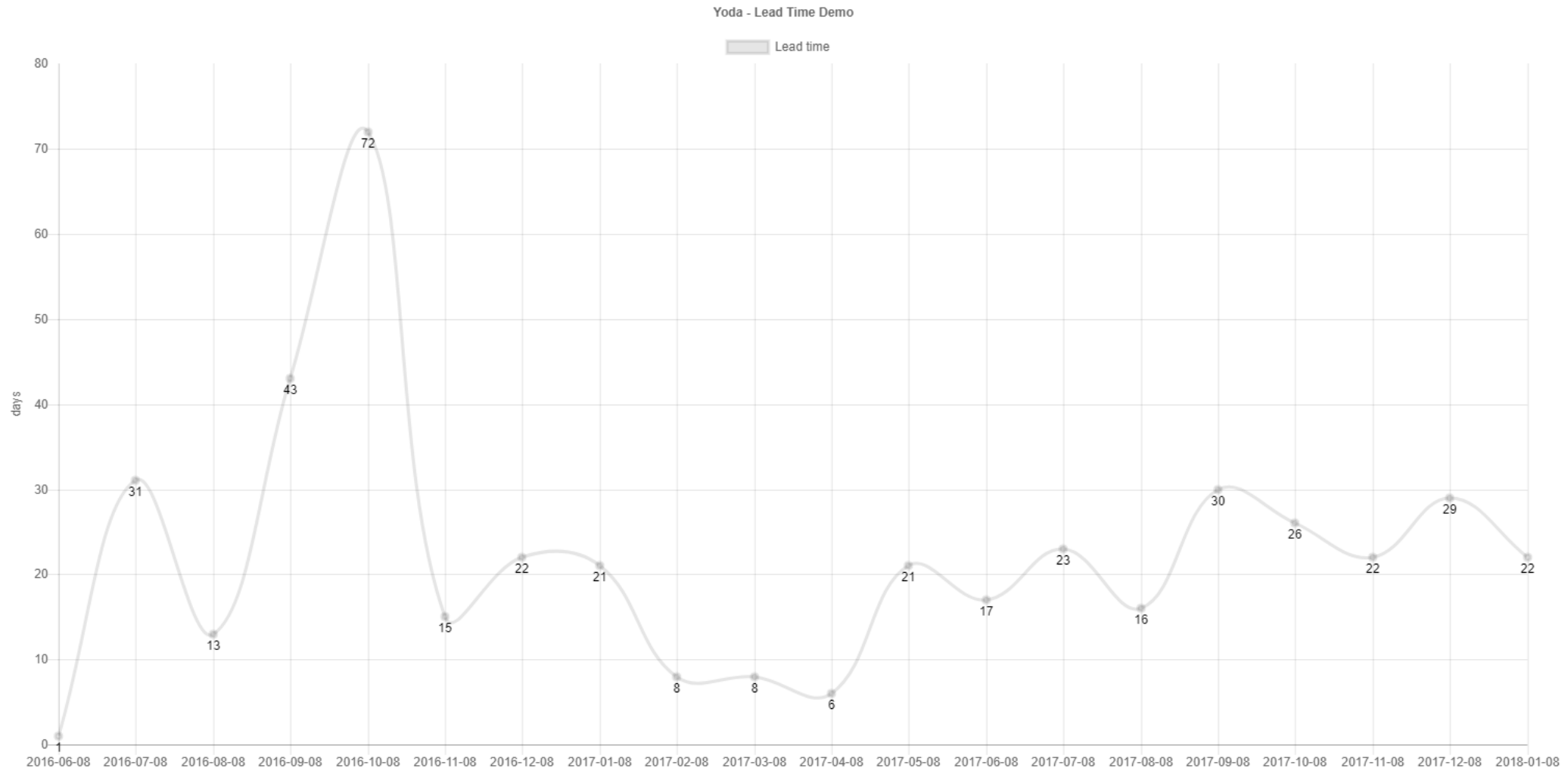
Owner	Repo	GitHub user	GitHub token	Label filter
		jens-markussen	T1 - Defect

Start date (blank=since first issue)	End date (today=blank)	Interval	Title	Draw CFD	Draw Lead Time
		7	Yoda - CFD Demo		



Lead Time Example

Owner	Repo	GitHub user	GitHub token	Label filter	
		jens-markussen	T1 - Defect	
Start date (blank=since first issue)	End date (today=blank)	Interval	Title	Draw CFD	Draw Lead Time
		7	Yoda - Lead Time Demo		

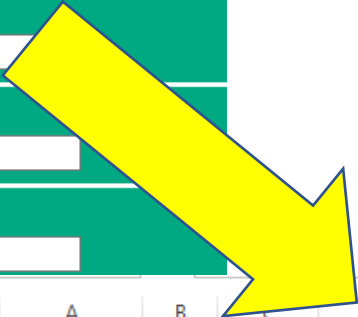


Issue Exporter

- Yoda Issue Exporter can **export issues** (all or filtered) to a **CSV file**, which can e.g. be **imported** into **Excel**.
- Exporter can export from a **single repo**, or across repos for an **entire GitHub Organization**.
- Set of exported **fields** are **highly configurable**.
- The use of a good **labelling convention** helps (e.g. as the tool supports merging Severity labels into a **single column**).

Issue Exporter Example

Owner	Repo	GitHub user	GitHub token	Label filter
github-yoda	demo	github-jens	
Multi-label column definitions				
Severity=^S[1-4] -,Issue Type=^T[1-9] -				
Single label column definitions (fiels automatically added to the end)				
Support				
Single label column regexps (fields automatically added to the end)				
Fields (further fields are: Body,Report Date, URL)				
Owner,Repo,Number,Issue Type,Severity,State,Submitter,Assignee,Milestone,Created at,Closed at,Duration,Title,Estimate,Remaining				
CSV delimiter	Label indicator	Issue state	Output file name	
;	1	all	issues.csv	
<input type="button" value="Export"/>				
Console				
Info: Initiating Github request: https://api.github.com/repos/github-yoda/demo/issues?state=all&di				
Info: Received 48 issues. Now analyzing and converting to CSV.				
Info: Data succesfully exported.				



	A	B	C	D	E	F	G	H	I	J
1	Owner	Repo	Number	Issue Type	Severity	State	Submitter	Assignee	Milestone	Created at
2	github-yoda	demo	1	T1 - Defect	S2 - High	open	hub-jens		Sprint 2	04-01-2018
3	github-yoda	demo	2	T3 - Task	S1 - Urgent	closed	hub-jens		Sprint 2	04-01-2018
4	github-yoda	demo	3	T1 - Defect	S1 - Urgent	closed	hub-jens			04-01-2018
5	github-yoda	demo	4	T2 - Enhancement	S4 - Low	open	hub-jens		Sprint 2	04-01-2018
6	github-yoda	demo	5	T1 - Defect	S2 - High	open	hub-jens		Sprint 2	04-01-2018
7	github-yoda	demo	6	T1 - Defect	S4 - Low	closed	hub-jens		Sprint 2	04-01-2018
8	github-yoda	demo	7	T1 - Defect	S2 - High	closed	hub-jens			04-01-2018
9	github-yoda	demo	8	T1 - Defect	S1 - Urgent	open	hub-jens			04-01-2018
10	github-yoda	demo	9	T1 - Defect	S2 - High	closed	hub-jens			04-01-2018
11	github-yoda	demo	10	T1 - Defect	S3 - Medium	open	hub-jens		Sprint 2	04-01-2018
12	github-yoda	demo	11	T1 - Defect	S3 - Medium	open	hub-jens			04-01-2018
13	github-yoda	demo	12	T2 - Enhancement	S4 - Low	closed	hub-jens			04-01-2018
14	github-yoda	demo	13	T1 - Defect	S4 - Low	open	hub-jens		Sprint 2	04-01-2018
15	github-yoda	demo	14	T1 - Defect	S3 - Medium	closed	hub-jens		Sprint 2	04-01-2018
16	github-yoda	demo	15	T1 - Defect	S4 - Low	closed	hub-jens			04-01-2018
17	github-yoda	demo	16	T1 - Defect	S2 - Medium	closed	hub-jens		Sprint 2	04-01-2018

Yoda Agile Project Management Tools

Burndown Chart, Velocity Chart, Kanban Board

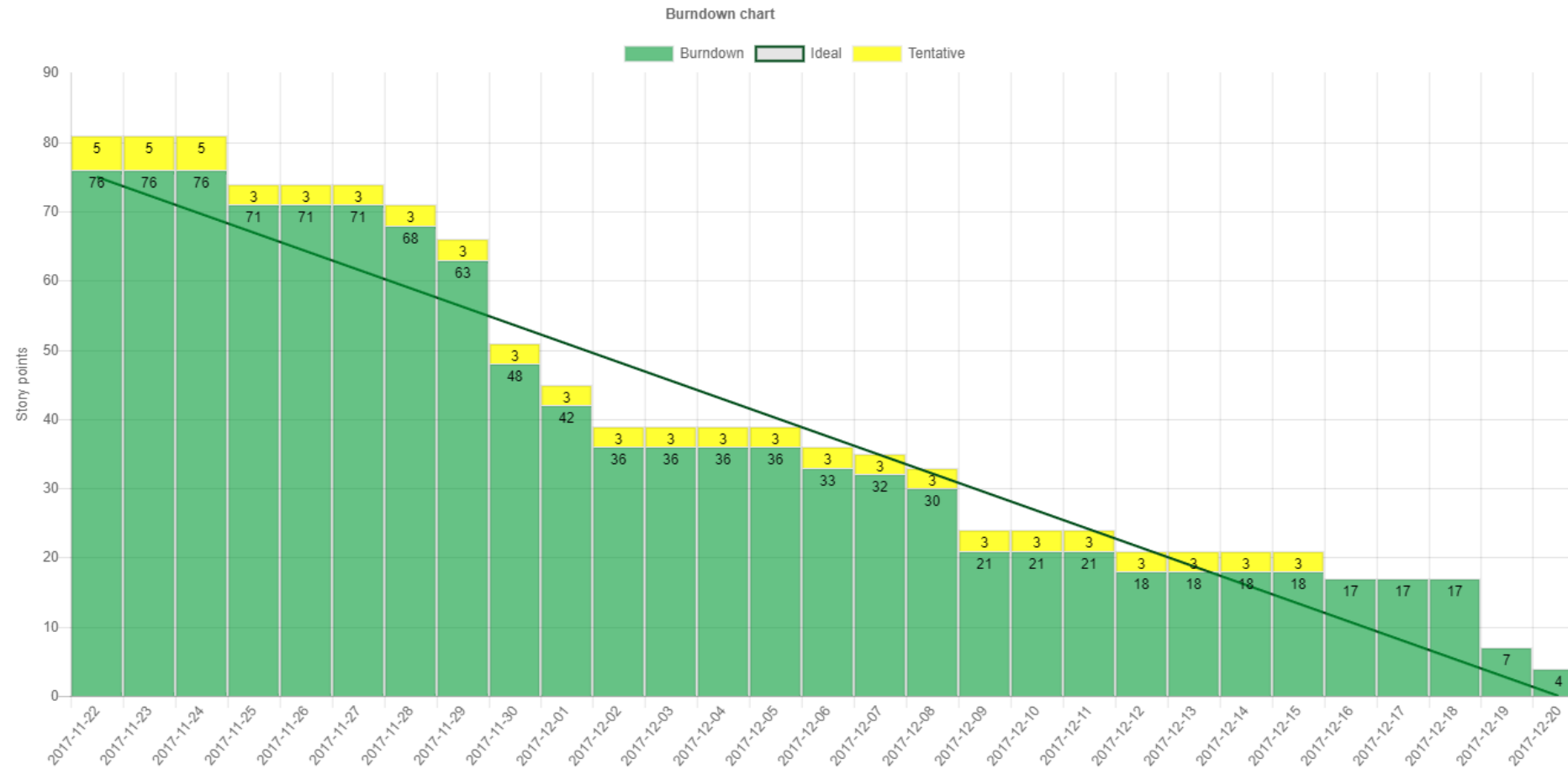
Burndown Chart

- A **Burndown Chart** is a **bar chart** showing the **remaining effort** over time for a given **sprint**
- An **ideal burndown** line is drawn for comparison
- Yoda uses **remaining estimates** (see earlier) for this purpose.
- It is possible to attribute some issues as **tentative** (aka **stretch goal**). These will be drawn in Yellow on top of committed issues
- Yoda Burndown tools further includes a table view containing the relevant sprint issues and their planning data.

Burndown Example

Owner	Repo	GitHub user	GitHub token	Tentative Label	In Progress Label	Label subtotals	Estimates
		jens-markussen	P - Tentative	Q - In Progress	^T[1-9] -	<input type="radio"/> # issues <input type="radio"/> In body <input type="radio"/> In Labels

Milestone	Project	Start date	Due date	Capacity	Show closed	Draw chart	Show table	Goto GitHub
Select milestone ...	Select project ...	2017-11-22	2017-12-20	75	<input checked="" type="checkbox"/>			



Burndown Table Example

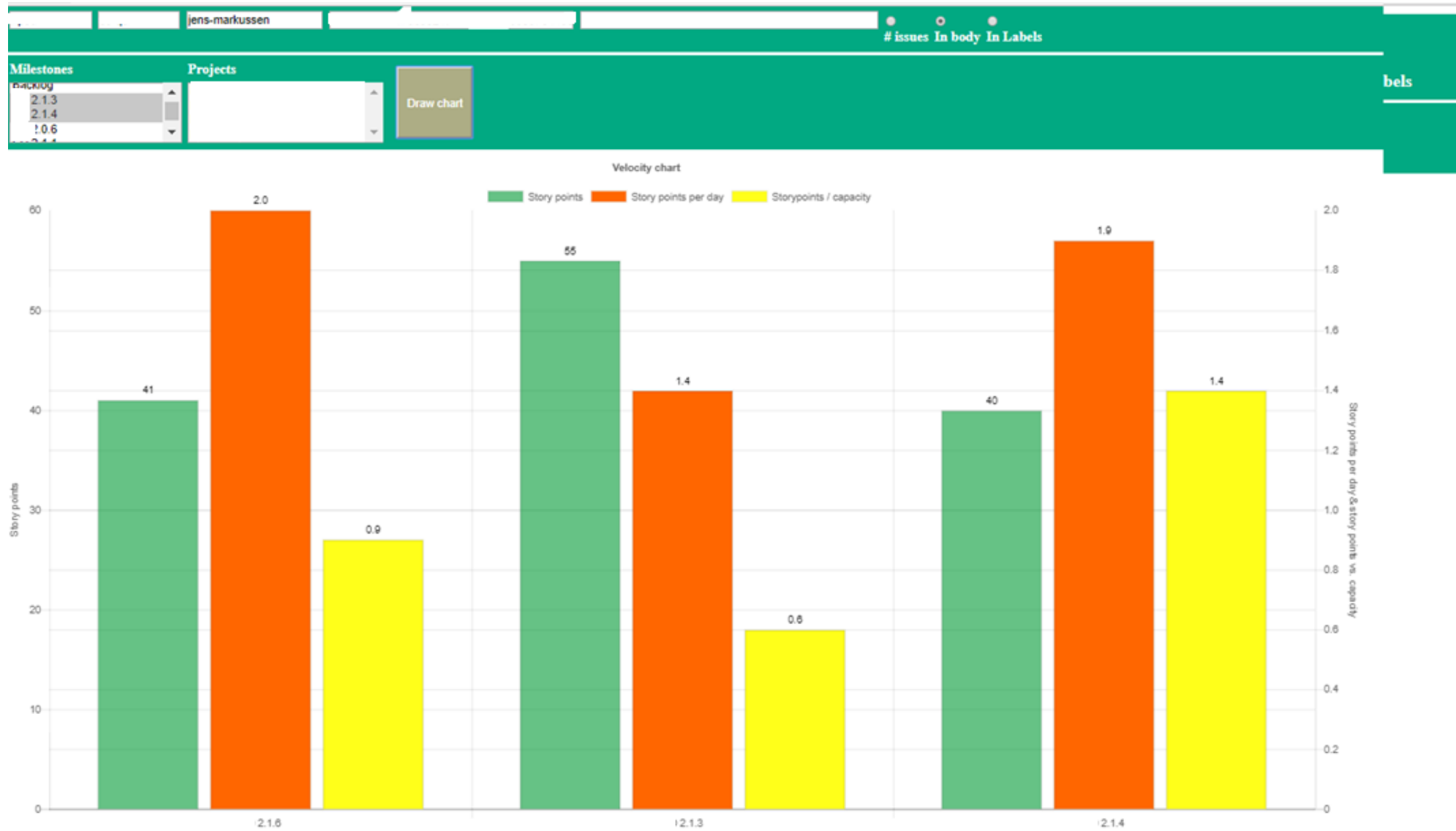
Owner	Repo	GitHub user	GitHub token	Tentative Label	In Progress Label	Label subtotals	Estimates	
github-yoda	demo	github-jens	P - Tentative	Q - In Progress	^T[1-9] -	<input type="radio"/> # issues <input checked="" type="radio"/> In body <input type="radio"/> In Labels	
Milestone	Project	Start date	Due date	Capacity	Show closed	Draw chart	Show table	Goto GitHub
Sprint 2	Select project ...	2018-03-01	2018-03-16	65	<input checked="" type="checkbox"/>			

Issue Id (11)	Assignee	Tentative?	Type	Issue Title	Estimate	Remaining	# Tasks	# Tasks done	Tentative	State
demo/1			T1 - Defect	Null pointer exception while trying to open web page	2	2	0	0	0	open
demo/2		Yes	T3 - Task	Enhance Yoda Kanban board with horizontal scroll bars	0	0	0	0	0	closed
demo/4		Yes	T2 - Enhancement	Prevent multiple editors in editor.	0	0	0	0	8	open
demo/5			T1 - Defect	Error drawing with mouse on Mondays.	5	5	0	0	0	open
demo/6			T1 - Defect	Problem updating topic list in editor.	6	0	0	0	0	closed
demo/10			T1 - Defect	Problem updating topic list during startup.	4	4	0	0	0	open
demo/13		Yes	T1 - Defect	Problem editing text on Mondays.	0	0	0	0	1	open
demo/14			T1 - Defect	Problem rebooting PC on Mondays.	3	0	0	0	0	closed
demo/16			T1 - Defect	Problem while creating new objects in Chrome browser.	3	0	0	0	0	closed
demo/18			T1 - Defect	Unknown error while creating new objects in Chrome browser.	3	0	0	0	0	closed
demo/20			T1 - Defect	Problem drawing with mouse in Chrome browser.	3	3	0	0	0	open
Grand Total					29	14	0	0	9	11
Subtotal	open				14	14	0	0	9	6
Subtotal	closed				15	0	0	0	0	5
Subtotal	In progress				0	0	0	0	0	0

Velocity Chart

- A **velocity chart** compares the team **velocity across** different **sprints**
- Over time, a velocity chart will **help teams** to set the **correct capacity** for upcoming sprints
- Yoda does this by reporting per sprint
 - number of **story points** completed
 - story points **per day**
 - story points **vs.** predefined sprint **capacity**

Velocity Chart Example



Kanban Board

- A **Kanban Board** shows **sprint activities** across various states, thus allowing an intuitive view of progress
- **GitHub** natively supports Kanban Boards as part of **projects**, where issues can be placed in configurable columns
- **Yoda** does not use this mechanism, but instead supports **Kanban Board** views of issues based on **issue labels** (e.g. Severities, defined Sub-states, issue types)
- **Issues** may be further **filtered** based on **milestones, labels, and assignee**
- Yoda Kanban boards can include **issues** from **multiple repos** inside the same organization.
- **Drag and drop** between columns **change labels** and can close (or reopen) issues
- **Note:** While Yoda Kanban boards provides label and state (open/closed) consistency, GitHub projects do not. Here issue to column is manually maintained.

Kanban Board Example

Owner

github-yoda

Github user

github-jens

Github token

.....

Columns

[Defect]open:T1 - Defect,[Enhancement]open:T2 - Enhancem...

Closed milestones

☐

Closed issues

☒

Locked

☐

Estimates

☐ # issues ☐ In body ☐ In

Repositories

× demo

Milestones (filter)

× Sprint 2

Labels

Assignees

Defect8 (20/20)

Issue editing text during weekends.
[demo#47 Sprint 2](#) *unassigned*
3 S3 - Medium T1 - Defect

Error drawing with mouse during weekends.
[demo#46 Sprint 2](#) *unassigned* (1)
S1 - Urgent T1 - Defect

Issue editing text in Chrome browser.
[demo#38 Sprint 2](#) *unassigned* (4)
S4 - Low T1 - Defect

Issue updating topic list during weekends.
[demo#36 Sprint 2](#) *unassigned* (1)
S3 - Medium T1 - Defect

Problem drawing with mouse in Chrome browser.
[demo#20 Sprint 2](#) *unassigned* (3)
S1 - Urgent T1 - Defect

Problem updating topic list during startup.
[demo#10 Sprint 2](#) *unassigned* (4)
S3 - Medium T1 - Defect

Error drawing with mouse on Mondays.
[demo#5 Sprint 2](#) *unassigned* (5)
S1 - Urgent T1 - Defect

Enhancement3 (10/10)

Unknown error editing text in editor.
[demo#43 Sprint 2](#) *unassigned* (1)
S2 - High T2 - Enhancement

Problem editing text on Mondays.
[demo#13 Sprint 2](#) *unassigned* (1)
P - Tentative S4 - Low T2 - Enhancement

Prevent multiple editors in editor.
[demo#4 Sprint 2](#) *unassigned* (8)
P - Tentative S4 - Low T2 - Enhancement

Task1 (5/5)

Update firmware in Safari browser.
[demo#41 Sprint 2](#) *unassigned* (5)
S1 - Urgent T3 - Task

Closed5 (15/15)

Unknown error while creating new objects in Chrome browser.
[demo#18 Sprint 2](#) *unassigned* (3)
S2 - High T1 - Defect

Problem while creating new objects in Chrome browser.
[demo#16 Sprint 2](#) *unassigned* (3)
S3 - Medium T1 - Defect

Problem rebooting PC on Mondays.
[demo#14 Sprint 2](#) *unassigned* (3)
S3 - Medium T1 - Defect

Problem updating topic list in editor.
[demo#6 Sprint 2](#) *unassigned* (6)
S4 - Low T1 - Defect

Enhance Yoda Kanban board with horizontal scroll bars
[demo#2 Sprint 2](#) *unassigned*
P - Tentative S1 - Urgent T3 - Task

Other Yoda tools

Label Manager, Admin, Task Copy

Label Manager

- In support of managing **labelling conventions** across **different repos**, Yoda includes a **label manager**
- The label manager can **copy labels** (all or some) from **one repo** to **another**.
- Label manager does **not allow deletion** of labels that are **in use**
- **Hint:** When creating a new repo, press "Delete all labels" to get rid of the standard GitHub labels. Next press "Copy all Labels" to get label definitions from your favorite repo.

Label Manager Example

GitHub user

GitHub token

Source

Owner

Repo

Labels

Copy all labels

Goto github

Refresh Labels

Click to copy/update update to destination (left to right).

1

13

2

20

3

40

5

8

P - Tentative

Q - Recurring

S1 - Urgent

S2 - High

S3 - Medium

S4 - Low

T1 - Defect

T2 - Enhancement

T3 - Task

Destination

Owner

Repo

Labels

Delete all labels

Goto github

Refresh Labels

Click label to delete it (only if no issues using them). Delete all button to do same for all

bug

duplicate

enhancement

good first issue

help wanted

invalid

question

wontfix

Admin

- The Yoda **admin tool** allows the user to **store** various **defaults** into the **browser settings** (localStorage)
- Most notably, the **GitHub userId** and personal **access token** should be set here.

Yoda Admin Example

GitHub user and token	GitHub user github-jens	GitHub token [REDACTED]	Update token	Delete token
GitHub URL overwrites	GitHub API URL https://api.github.com/	GitHub HTML URL https://www.github.com/	Set github.com values	
Global Yoda defaults	Owner default github-yoda	Repo default demo	Estimates <input type="radio"/> # issues <input type="radio"/> In body <input type="radio"/> In Labels <input checked="" type="radio"/> (no default)	
Time Statistics defaults Overwrites	Interval [REDACTED]	Label Bar Splitting [REDACTED]	Other ("blank" for blank) [REDACTED]	
Burndown defaults overwrites	Tentative Label [REDACTED]	In Progress Label [REDACTED]	Label subtotals [REDACTED]	
CFD defaults overwrites	Interval [REDACTED]			
Kanban defaults overwrites	Columns [REDACTED]			

Task Copy

- When executing **successive sprints**, you may have **recurring tasks** that you need to execute for **every sprint**.
- These **tasks** should naturally be handled (including estimates) as GitHub **issues**.
- The **task copy tool** allows you to copy such tasks **from one sprint** (milestone) **to the next**.
- If such recurring issues include **tasks lists** (GitHub notation “– [x] text”), **check boxes** will be **cleared** in preparation for the next sprint (so “- [x]” will become “- []”)

Task Copy

GitHub user

github-jens

GitHub token

●●●●●●●●●●●●●●●●●●●●

Recurring Label Filter

Q - Recurring,T3 - Task

Body remove regexp

<^> remaining .*\$/,- \[(x|X)\] /- []

Bracket Title Copy

☒

Goto GitHub

Src Owner

github-yoda

Source Repo

demo

Source Milestone

Sprint 1

Dst Owner

github-yoda

Destination Repo

demo

Destination Milestone

Sprint 2

Refresh issues

Copy issues

Console

```
Getting source issues using URL: https://api.github.com/repos/github-yoda/demo/issues?state=all&direction=asc&labels=Q - Recurring,T3 - Task&milestone=2
Getting destination issues using URL: https://api.github.com/repos/github-yoda/demo/issues?state=all&direction=asc&labels=Q - Recurring,T3 - Task&milestone=1
Retrieved 0 destination issues.
Retrieved 2 source issues.
45: [Sprint 1] Refill coke machine for developers
    OK: This issue will be copied.
48: [Sprint 1] Clear log files ahead of new sprint.
    OK: This issue will be copied.
A total of 2 issues are ready to be copied.
```

Yoda Architecture

Yoda Architecture

- Yoda has a very **simple architecture** based on a few **key principles**:
 1. **All data** will be kept **in GitHub** – no auxiliary database will be used
 2. Yoda **executes** exclusively in the **browser**. Yoda has **no backend**, apart from GitHub.
 3. Yoda **communicates** with GitHub using the **standard API** (version 3)
 4. Yoda tools are written using only **HTML** and **JavaScript**
 5. Yoda uses various **JavaScript libraries**, which are all pulled from the Internet at cdn.com.
- Other **key features**:
 - Yoda can run against any the default **github.com** instance or against any **GitHub Enterprise** instance

Thank You