

Yoda

Agile Project Management with GitHub

Jens Vedel Markussen, Engineering Manager
Hewlett Packard Enterprise

Introduction

Yoda was developed during 2017/2018 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** (for both bugs and new 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
 - Milestone Manager, 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)
- Milestones with matching **titles across repositories** are considered to be part of the **same sprint**.
 - This allows **multi-repository** sprint **planning** and **tracking**

github-yoda / demo

<> Code Issues 29 Pull request

Labels Milestones

Title

Sprint 2

Due date (optional)

16-03-2018

Description

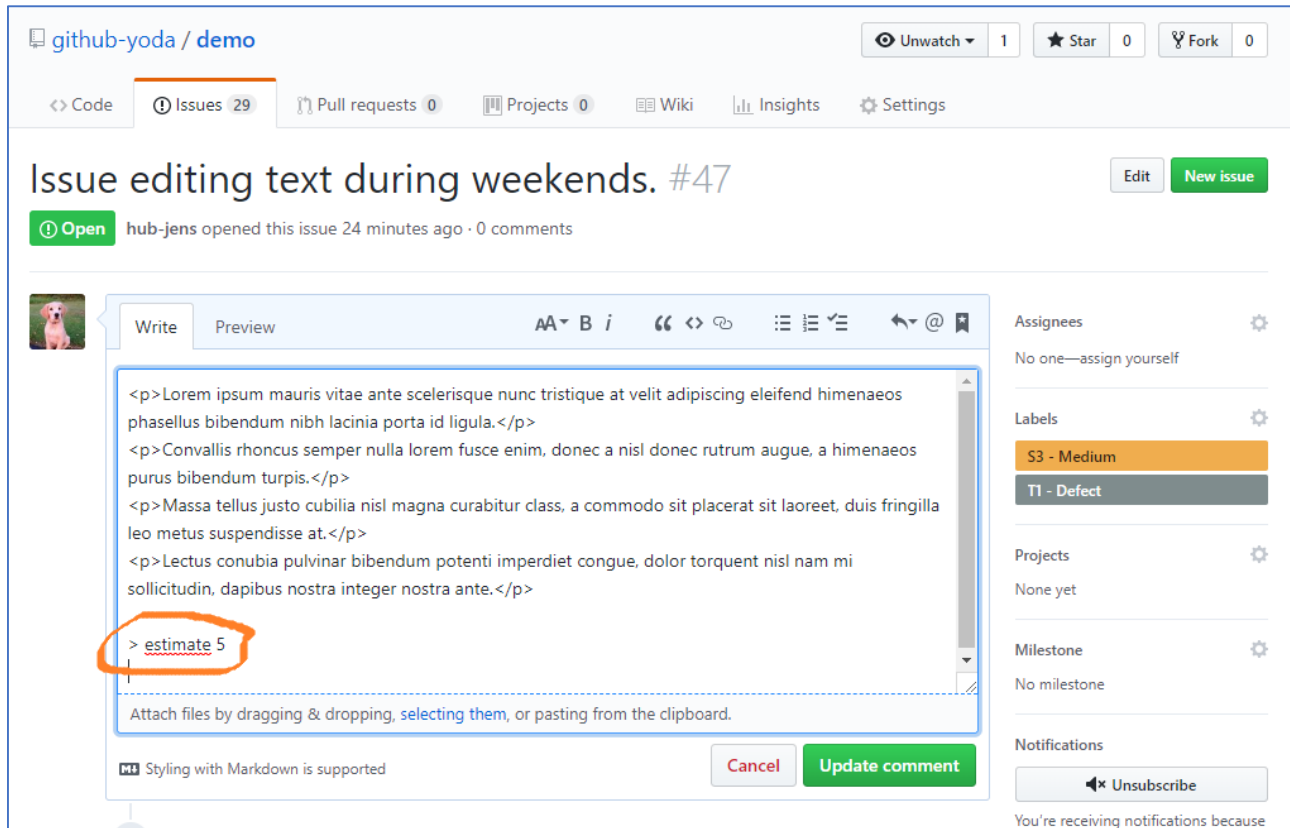
```
> startdate 2018-03-01
> capacity 65
|
```

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.)
 - This seems an **obvious omission** from GitHub
- 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 pre-defined 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)



github-yoda / demo

<> Code Issues 29 Pull requests 0 Projects 0 Wiki Insights Settings

Issue editing text during weekends. #47

Open hub-jens opened this issue 24 minutes ago · 0 comments

Write Preview

AA B i “ <> ↺ ⋮ ⋮ ⋮ ↶ @ 📎

<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>

> estimate 5

Attach files by dragging & dropping, selecting them, or pasting from the clipboard.

Styling with Markdown is supported

Cancel Update comment

Assignees
No one—assign yourself

Labels
S3 - Medium
T1 - Defect

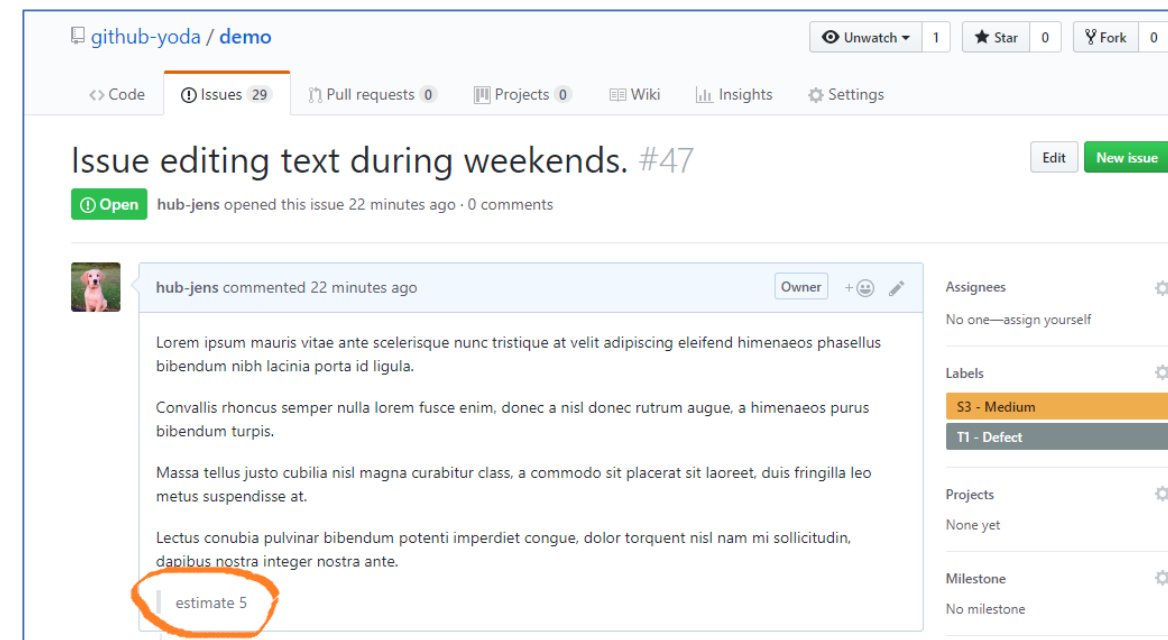
Projects
None yet

Milestone
No milestone

Notifications
Unsubscribe

You're receiving notifications because

Resulting preview/HTML



github-yoda / demo

<> Code Issues 29 Pull requests 0 Projects 0 Wiki Insights Settings

Issue editing text during weekends. #47

Open hub-jens opened this issue 22 minutes ago · 0 comments

hub-jens commented 22 minutes ago

Owner + 👤 ✎

Assignees
No one—assign yourself

Labels
S3 - Medium
T1 - Defect

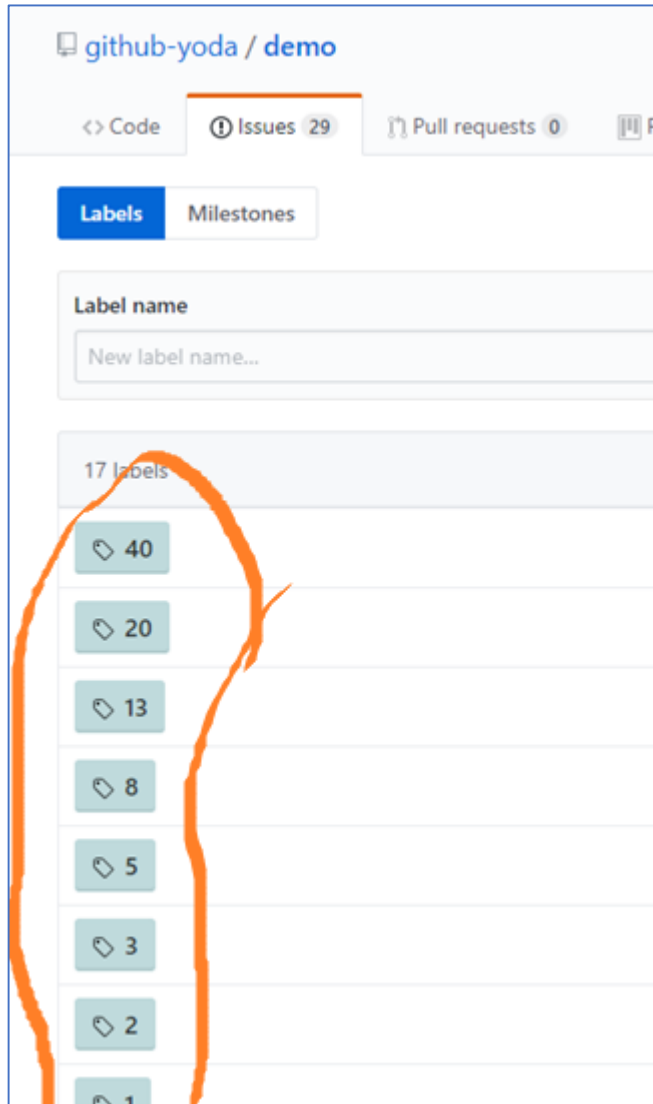
Projects
None yet

Milestone
No milestone

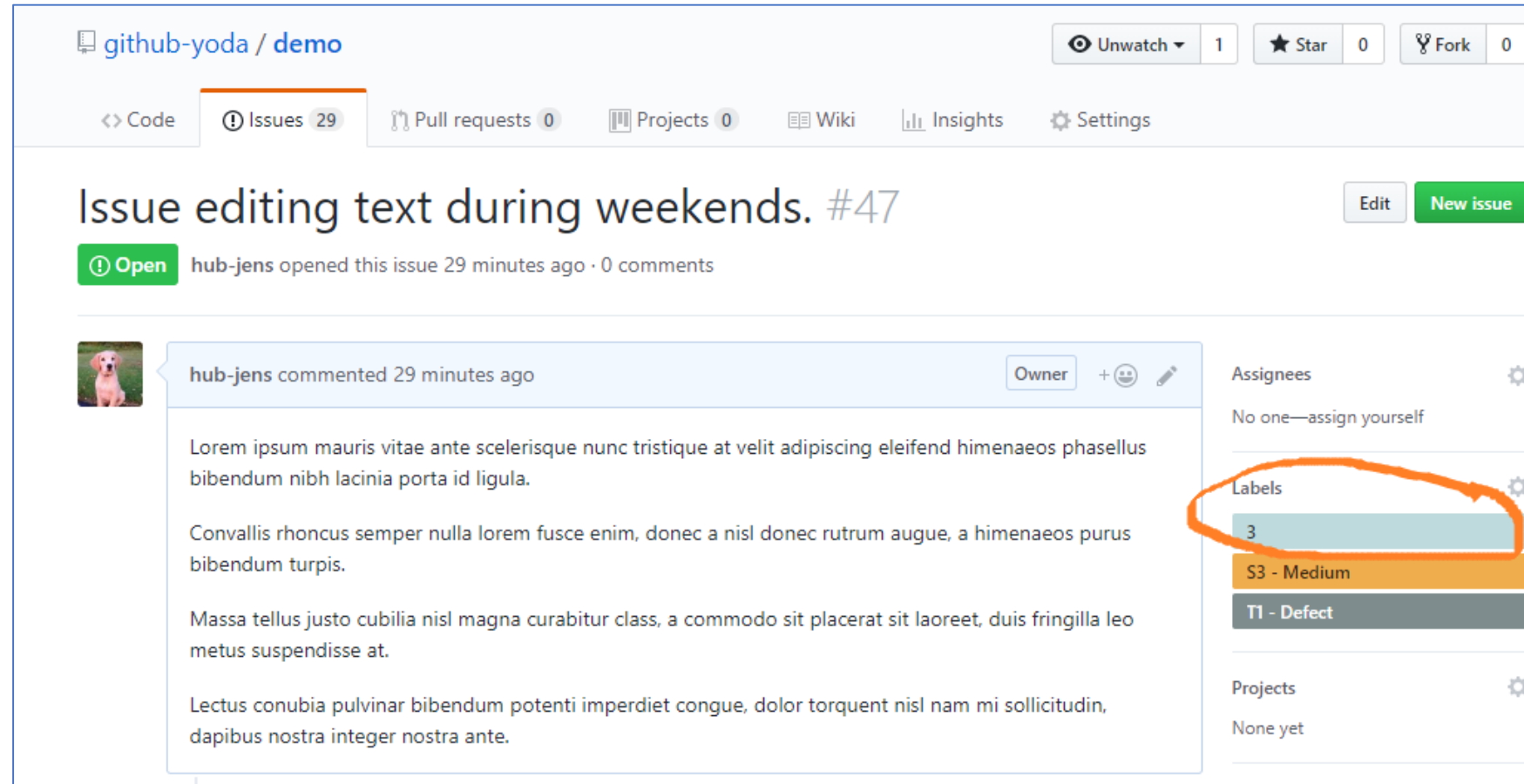
estimate 5

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 rich text editor with a toolbar. The content of the editor is as follows:

<p>Lorem ipsum orci dictumst fermentum luctus gravida, taciti ultricies bibendum dui vel platea, urna leo vivamus habitasse donec.</p>
<p>Vulputate convallis eget laoreet neque vulputate pharetra etiam purus quis, ad diam potenti elementum rhoncus rutrum aptent maecenas.</p>
<p>Luctus tempor integer inceptos facilisis vulputate amet magna tincidunt mi lacus, lectus porta quam viverra fames platea justo ante.</p>
<p>Aliquam sagittis nullam consectetur tempor tristique habitasse hendrerit posuere, lacus class sem ullamcorper aliquet orci.</p>

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

At the bottom of the editor, there is a note: "Attach files by dragging & dropping, selecting them, or pasting from the clipboard." and a status bar that says "Styling with Markdown is supported". There are "Cancel" and "Update comment" buttons at the bottom right.

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. There is 1 comment. The comment is from "hub-jens" and was posted 2 days ago. The comment content is as follows:

Lorem ipsum orci dictumst fermentum luctus gravida, taciti ultricies bibendum dui vel platea, urna leo vivamus habitasse donec.

Vulputate convallis eget laoreet neque vulputate pharetra etiam purus quis, ad diam potenti elementum rhoncus rutrum aptent maecenas.

Luctus tempor integer inceptos facilisis vulputate amet magna tincidunt mi lacus, lectus porta quam viverra fames platea justo ante.

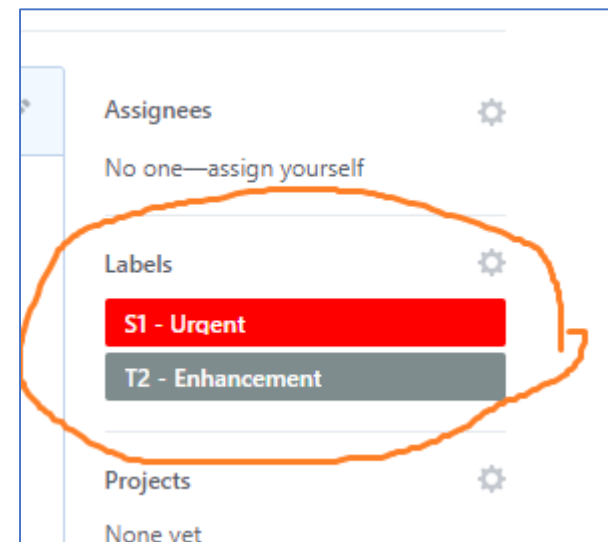
Aliquam sagittis nullam consectetur tempor tristique habitasse hendrerit posuere, lacus class sem ullamcorper aliquet orci.

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

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**
- **Scope** can be issues in the **entire organization**, or **one-** or **multiple** repositories
- 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

hewlettpackard

Repositories

x yoda-demo

Label filter

T1 - Defect

Count

☒ Issues ☐ Days open ☐ Opened ☐ Closed

Start date (blank=2m ago)

2017-12-29

End date (today=blank)

Interval

7

Label Bar Splitting

^S[1-4] -

Other (blank to omit)

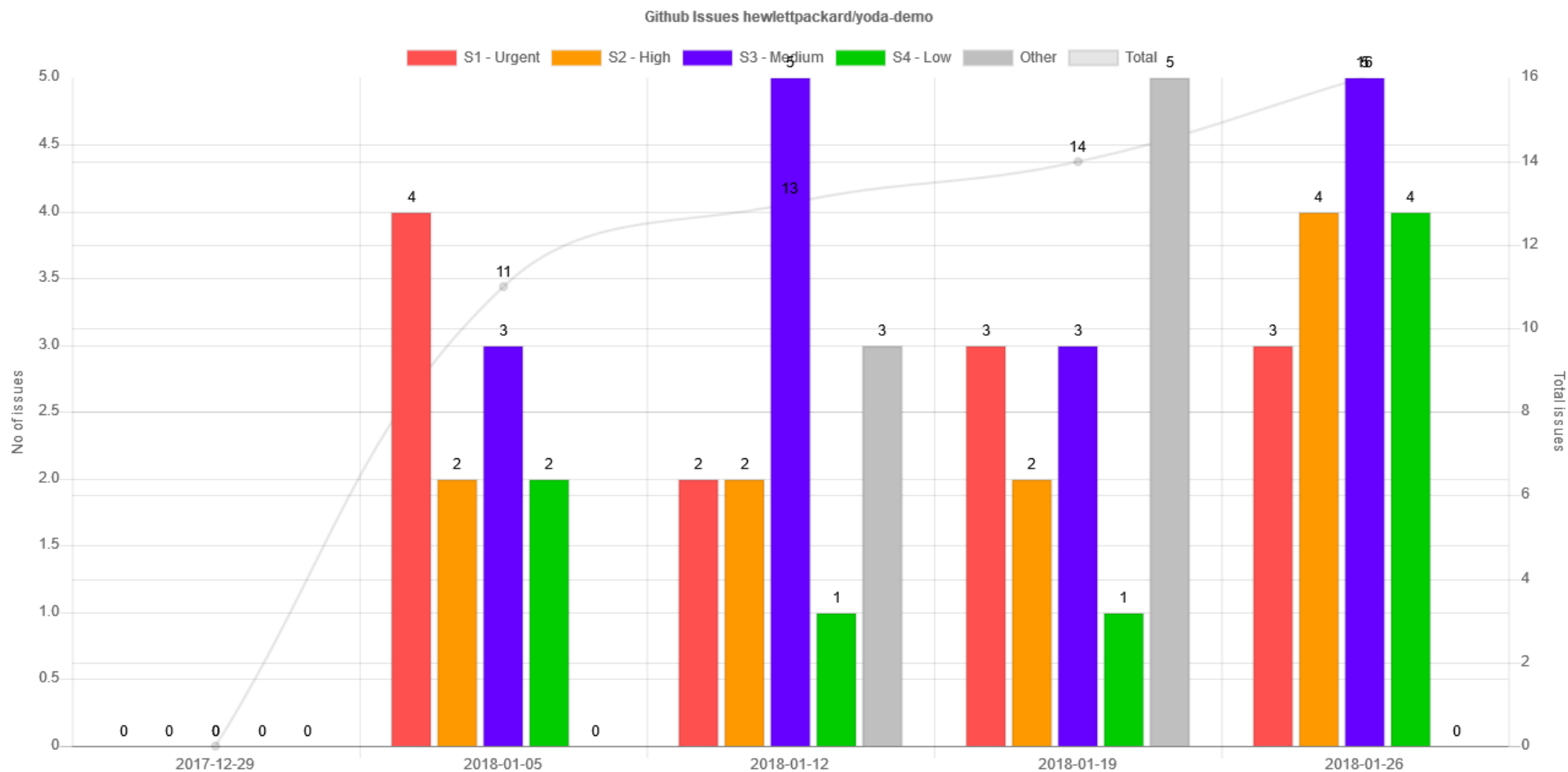
Other

Title

Stacked

☐

Draw chart



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**.
- **Scope** can be issues in the **entire organization**, or **one- or multiple** repositories
- 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

Repositories

Label filter

T2 - Enhancement

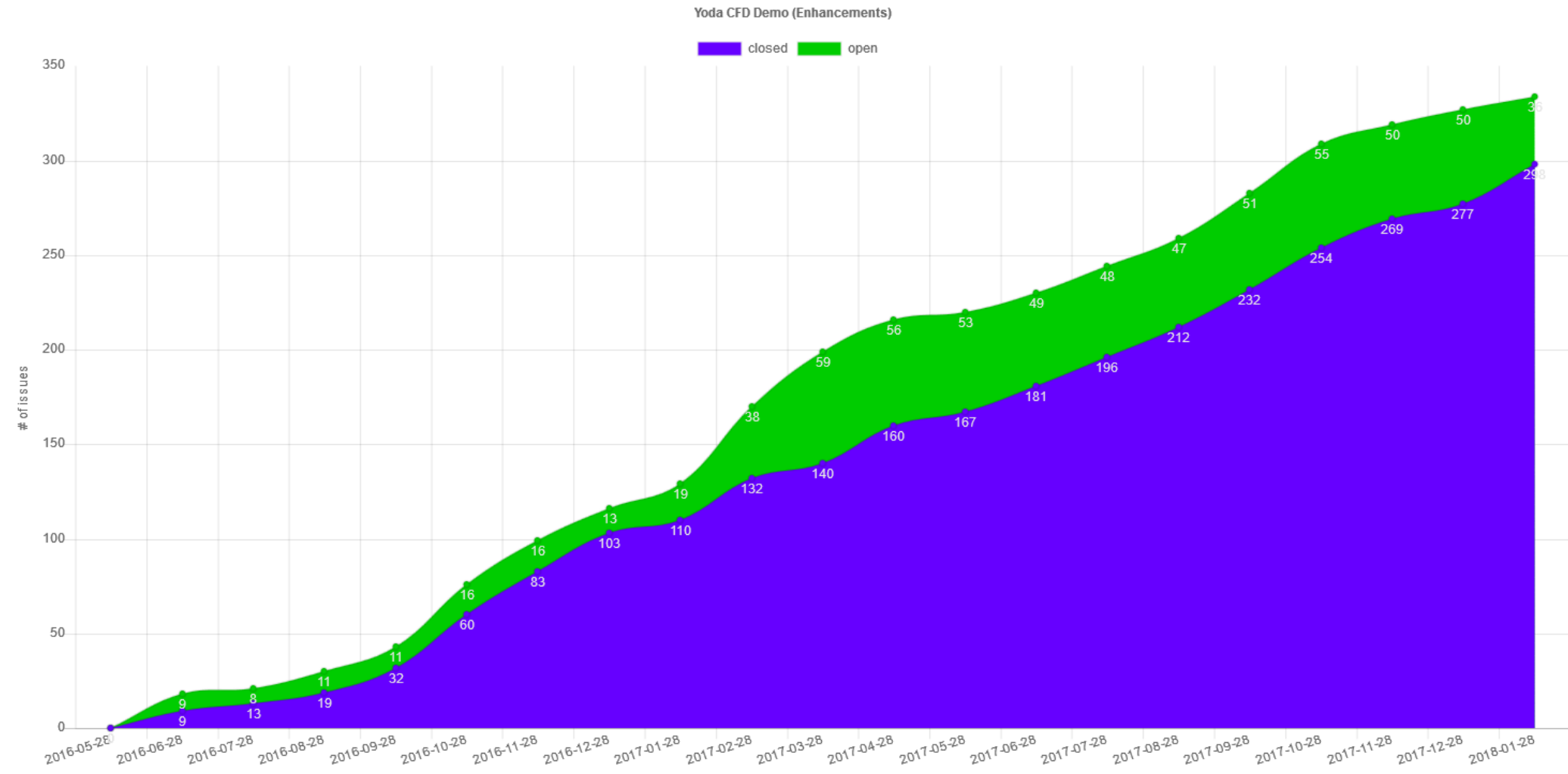
Start date (blank=since first issue) End date (today=blank) Interval Title

7

Yoda CFD Demo (Enhancements)

Draw CFD

Draw Lead Time



Lead Time Example

Owner

Repositories

Label filter

T2 - Enhancement

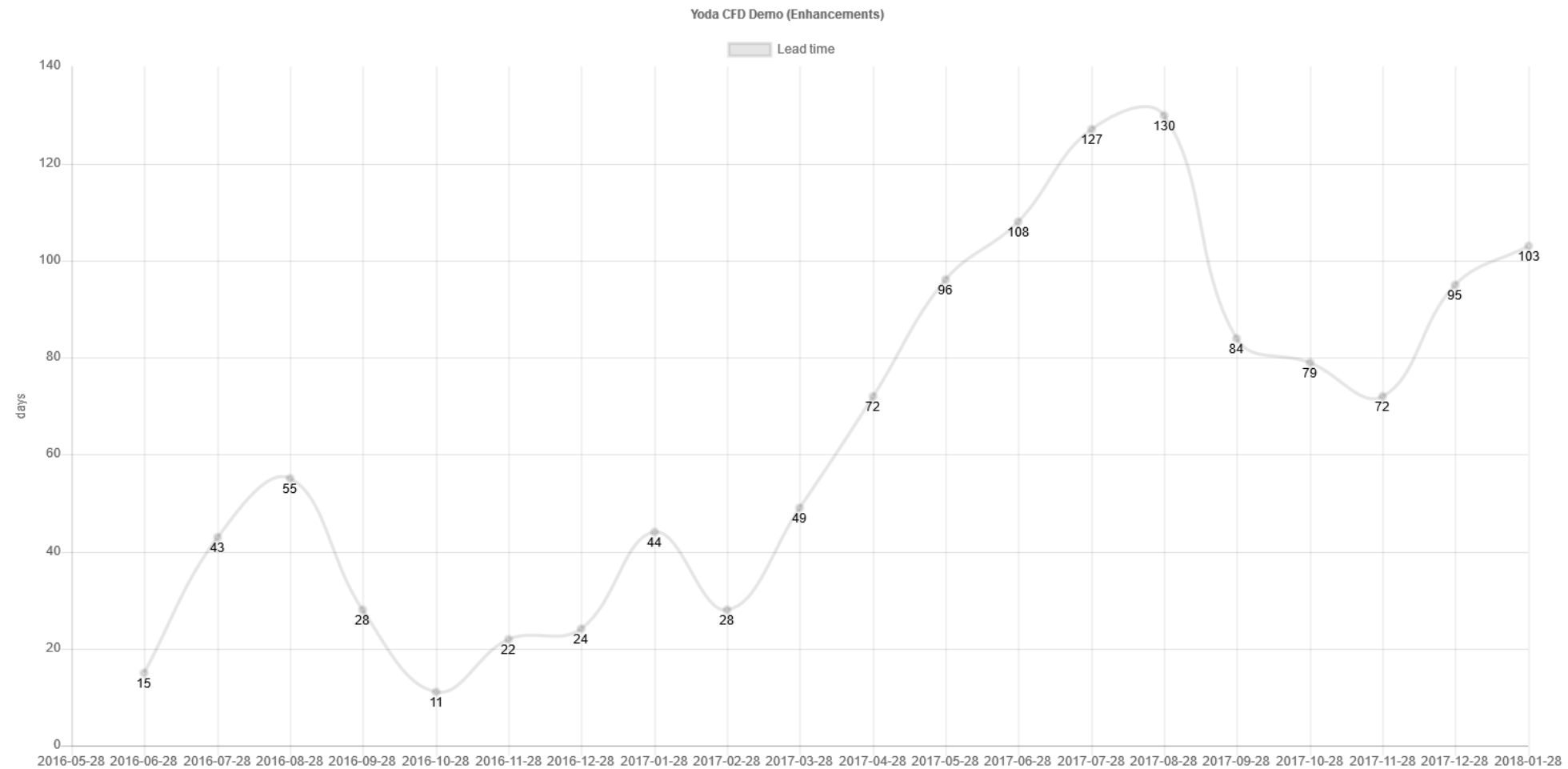
Start date (blank=since first issue)End date (today=blank)IntervalTitle

7

Yoda CFD Demo (Enhancements)

Draw CFD

Draw Lead Time



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**, **multiple repositories** or across all 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

Repositories

Label filter

hewlettpackard

×

yoda-demo

Multi-label column definitions

Severity=^S[1-4] -,Issue Type=^T[1-9] -

Single label column definitions (fiels automatically added to the end)

Support,Customer Encountered,P - Tentative

Single label column regexps (fields automatically added to the end)

^C -,^Th -

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

Estimates

Output file name

Export

;

1

all

☒ # issues

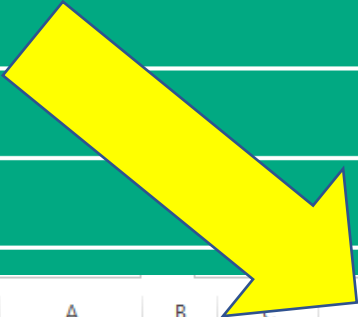
☐ In body

☐ In Labels

issues.csv

Console

Info: Initiated Github request.
Info: Received 118 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	C
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

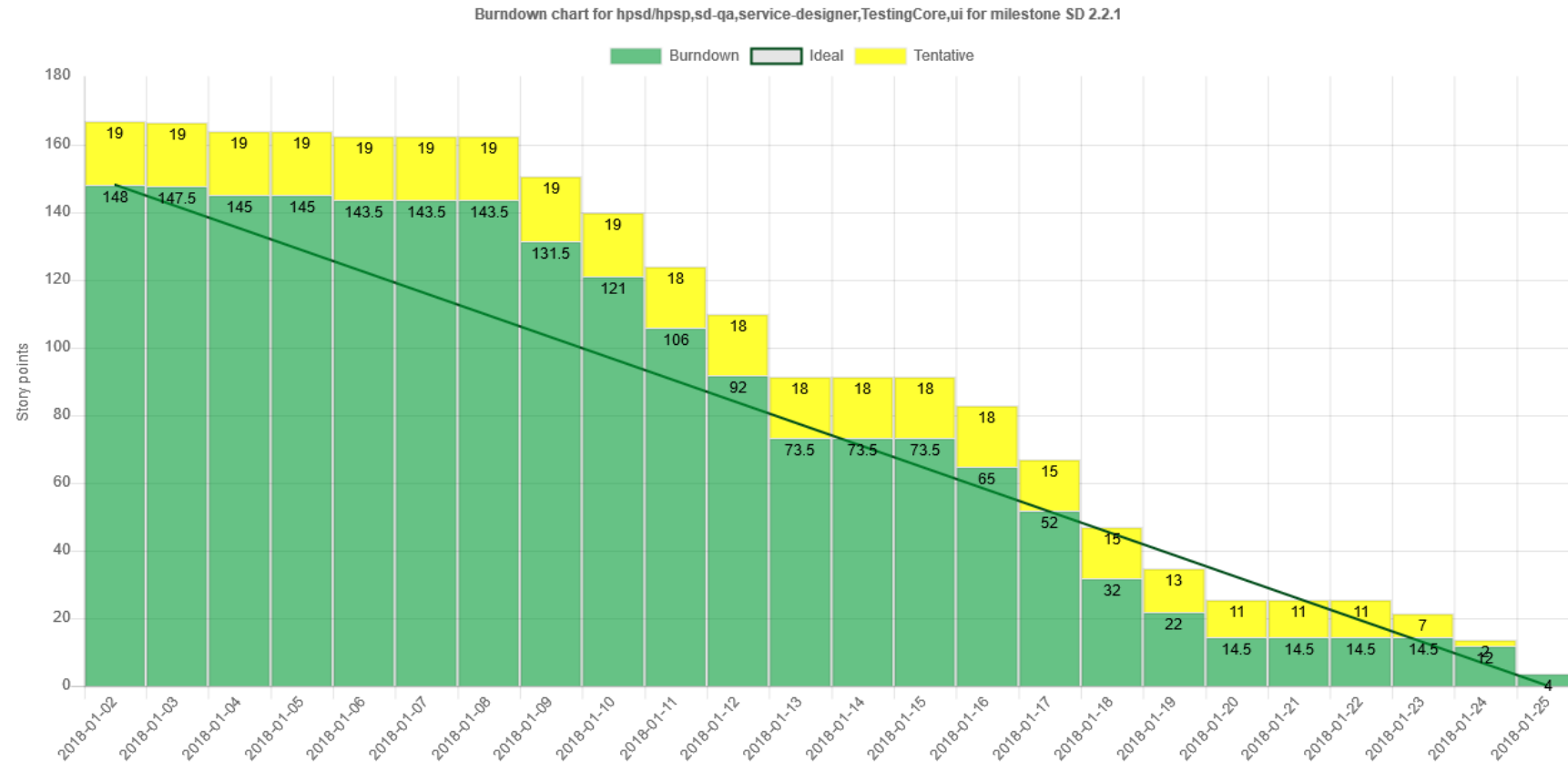
Burndown Chart, Velocity Chart, Kanban Board

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.
- **Scope** can be issues **one-** or **multiple** repositories. This allows **cross-repository planning** and **tracking**
- 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 I. _____	Tentative Label P - Tentative	In Progress Label Q - In Progress	Label subtotals ^T[1-9] -	Estimates <input type="radio"/> # issues <input checked="" type="radio"/> In body <input type="radio"/> In Labels	Closed issues <input checked="" type="checkbox"/>	Closed milestones <input checked="" type="checkbox"/>
Repositories _____		Milestone 2.2.1	Start date 2018-01-02	Due date 2018-01-25	Capacity _____	Draw chart	Show table



Burndown Table Example

Owner

Tentative Label

In Progress Label

Label subtotals

Estimates

Closed issues

Closed milestones

hewlettpackard

P - Tentative

Q - In Progress

^T[1-9] -

issues

In body

In Labels

Repositories

Milestone

Start date

Due date

Capacity

Draw chart

Show table

× yoda-demo

Sprint 1

Issue Id (10)	Assignee	Tentative?	Type	Issue Title	Estimate	Remaining	# Tasks	# Tasks done	Tentative	State
yoda-demo/42			T3 - Task	Unknown error updating topic list in Chrome browser.	6	0	0	0	0	closed
yoda-demo/44			T3 - Task	Problem editing text in Chrome browser.	3	3	0	0	0	open
yoda-demo/45			T3 - Task	[Sprint 1] Refill coke machine for developers	0	0	2	0	0	closed
yoda-demo/48			T3 - Task	[Sprint 1] Clear log files ahead of new sprint.	1	0	3	2	0	closed
yoda-demo/58			T1 - Defect	Issue drawing with mouse during weekends.	4	0	0	0	0	closed
yoda-demo/59			T2 - Enhancement	Discontinue system view on test systems.	4	0	0	0	0	closed
yoda-demo/64			T1 - Defect	Enhance boot process in editor.	7	7	0	0	0	open
yoda-demo/69			T1 - Defect	Issue updating topic list in Chrome browser.	5	0	0	0	0	closed
yoda-demo/70			T1 - Defect	Issue updating topic list during weekends.	5	5	0	0	0	open
yoda-demo/75			T2 - Enhancement	Allow boot process in editor.	6	0	0	0	0	closed
Grand Total					41	15	5	2	0	10
Subtotal	open				15	15	0	0	0	3
Subtotal	closed				26	0	5	2	0	7
Subtotal	In progress				0	0	0	0	0	0
Label subtotals										
Subtotal			T1 - Defect		21	12	0	0	0	4
Subtotal					12	12	0	0	0	2

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
- **Scope** can be issues **one-** or **multiple** repositories. This allows **cross-repository planning** and **tracking**
- 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

Owner

Repositories

Milestone

Estimates

Closed milestones

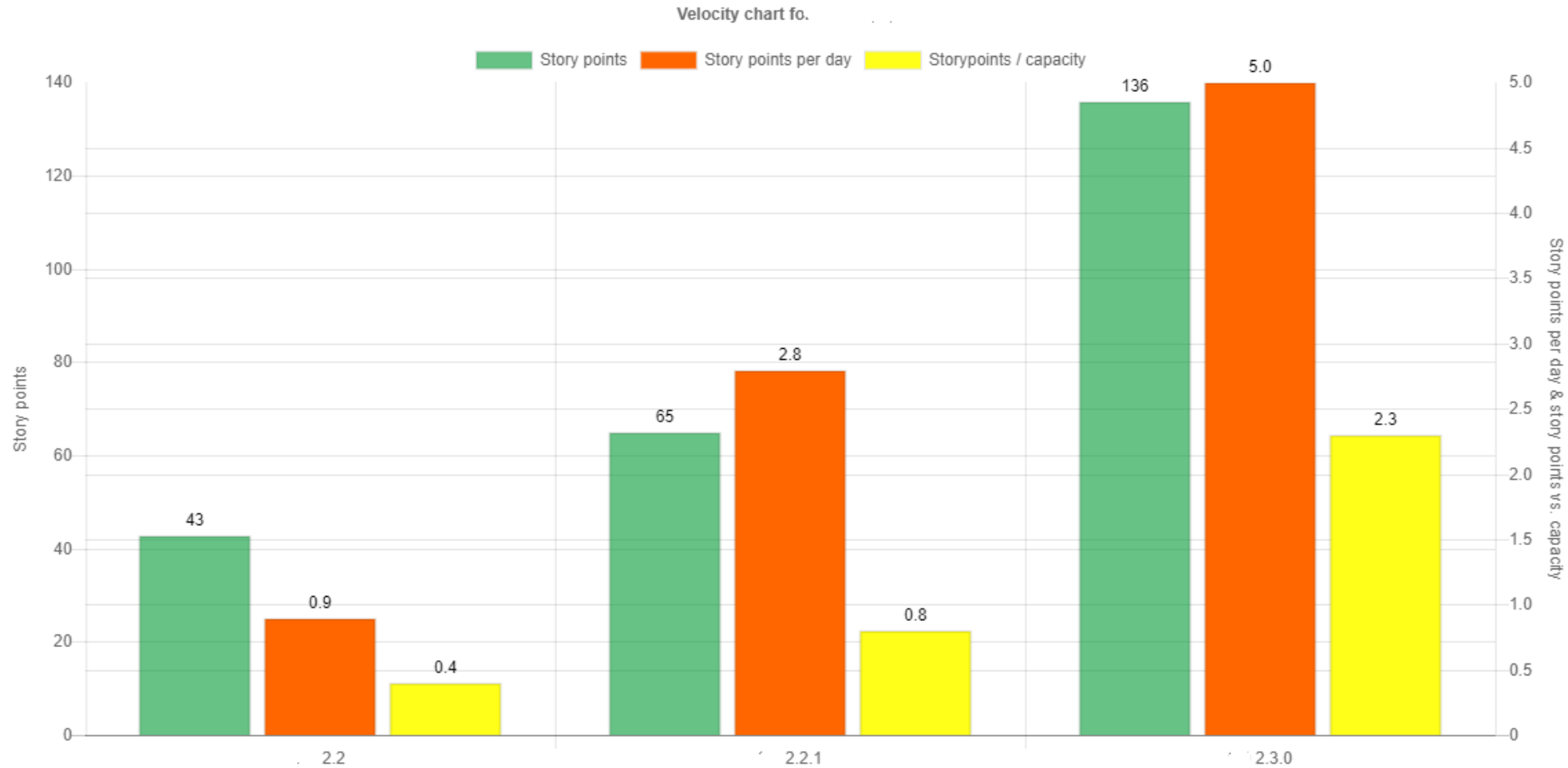
Draw chart

issues

In body

In Labels

☒



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

hewlettpackard

Columns

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

Closed milestones

☐

Closed issues

☒

Locked

☒

Estimates

☐ # issues ☐ In body ☐ In Labels

Repositories

× yoda-demo

× yoda-demo2

Milestones (filter)

× Sprint 1

× Sprint 2

Labels

Assignees

Other Yoda tools

Milestone Manager, Label Manager, Admin, Task Copy

Milestone Manager

- In support of managing **sprints** as a **set of milestones** across **different repos**, Yoda includes a **milestone manager**
- The milestone manager can **create milestones** automatically across several repositories
- Also, the tool can **synchronize sprint milestones** across repositories (updating e.g. due date in sync)

Milestone Manager Example

Owner

hewlettpackard

Repositories

× yoda

× yoda-demo

× yoda-demo2

Milestone(s)

× Sprint 1

Closed milestones

☐


Refresh

Repository	Milestone	State	Description	Start Date	Due Date	Burndown Date	Capacity (total 50)	Actions
All	<input type="text"/>		<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>		<div>Create milestone</div>
yoda-demo	Sprint 1	<div>open</div>	<div>Yoda sprint demo #1</div>	<div>2018-01-14</div>	<div>2018-02-28</div>	<input type="text"/>	<div>50</div>	<div>Copy/Update</div>
yoda-demo2	Sprint 1	<div>open</div>	<div>Yoda sprint demo #1</div>	<div>2018-01-14</div>	<div>2018-02-28</div>	<input type="text"/>	<div></div>	<div>Copy/Update</div>

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



Source

Owner

hewlettpackard

Repo

yoda-demo

Labels

18

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 - In Progress	Q - Recurring	S1 - Urgent
S2 - High	S3 - Medium	S4 - Low
T1 - Defect	T2 - Enhancement	T3 - Task

Destination

Owner

hewlettpackard

Repo

yoda-demo2

Labels

8

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 jens-markussen	GitHub token <secret>	Update token	Delete token
GitHub URL overwrites	GitHub API URL	GitHub HTML URL	Set github.com values	Set HPE GitHub values
Global Yoda defaults	Owner default hewlettpackard	Repolist default yoda-demo,yoda-demo2	Estimates <input checked="" type="radio"/> # issues <input checked="" type="radio"/> In body <input checked="" type="radio"/> In Labels <input type="radio"/> (no default)	
Time Statistics defaults overwrites	Interval	Label Bar Splitting	Other ("blank" for blank)	
Burndown defaults overwrites	Tentative Label	In Progress Label	Label subtotals	
CFD defaults overwrites	Interval			
Kanban defaults overwrites	Columns			

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 Example

Owner

hewlettpackard

Repositories

× yoda

× yoda-demo

× yoda-demo2

Milestone(s)

× Sprint 1

Closed milestones

☐

Refresh

Repository	Milestone	State	Description	Start Date	Due Date	Burndown Date	Capacity (total 50)	Actions
All	<input type="text"/>		<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>		Create milestone
yoda-demo	Sprint 1	open ▾	Yoda sprint demo #1	2018-01-14	2018-02-28	<input type="text"/>	50 <input type="text"/>	Copy/Update
yoda-demo2	Sprint 1	open ▾	Yoda sprint demo #1	2018-01-14	2018-02-28	<input type="text"/>	<input type="text"/>	Copy/Update

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