Atlassian

Rewriting a Plugin Architecture 3 Times to Harness the API Economy

A couple of Atlassians who are passionate about plugins

API Economy

Economy

$50,000,000

*Paid to ISVs to date.

15,000,000 paying users

*2011. http://techcrunch.com/2012/04/24/facebook-revenue-share-apps/

$1,000,000,000

*Paid to ISVs in 2013. http://www.partner-path.com/2013/12/salesforce-partner-ecosystem/

REST?

OAuth?

Blue logos?

SPI API Economy

API

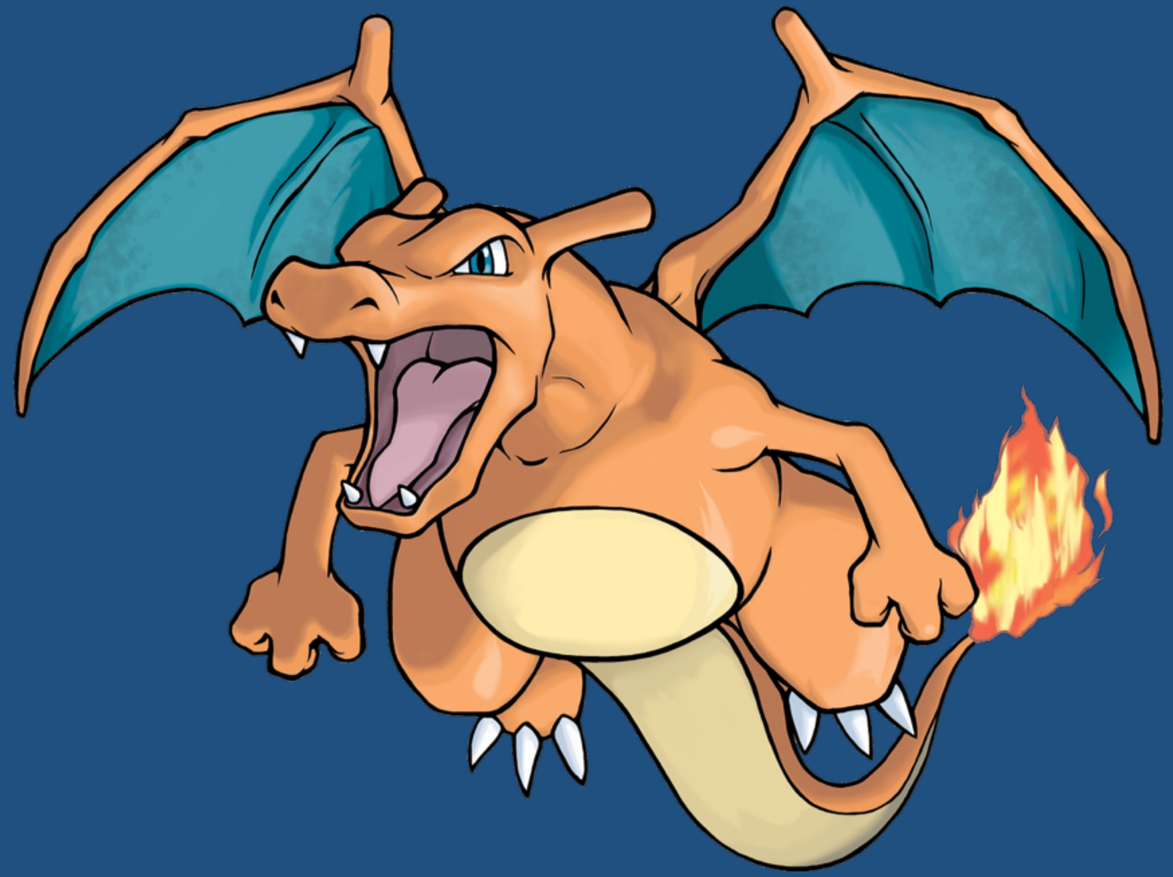
Application Programming Interface

SPI

Service Provider Interface

Atlassian Plugins (2004)

P2
(2008)



Atlassian Connect (2014)

Atlassian Plugins (2004)

Open(-ish) Source [2001]

Why it’s awesome!

Customers can “scratch their own itch”

Atlassian devs can focus on the 90%

Customization == stickier product

Devs who customize, evangelize

Why it sucks..

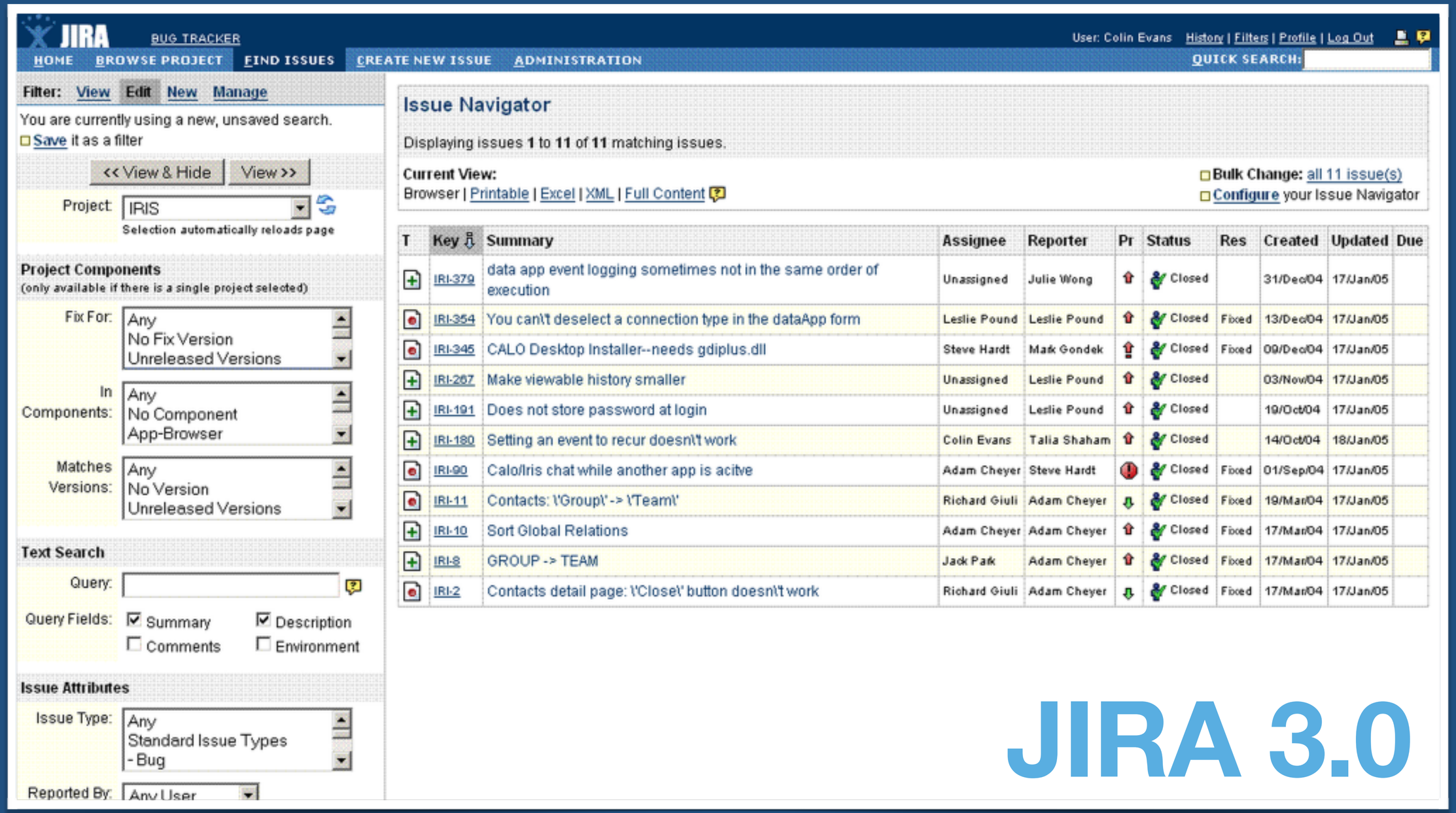
Massive barrier to upgrade

Pain to build the entire product

Poor plugin UX == poor product perception

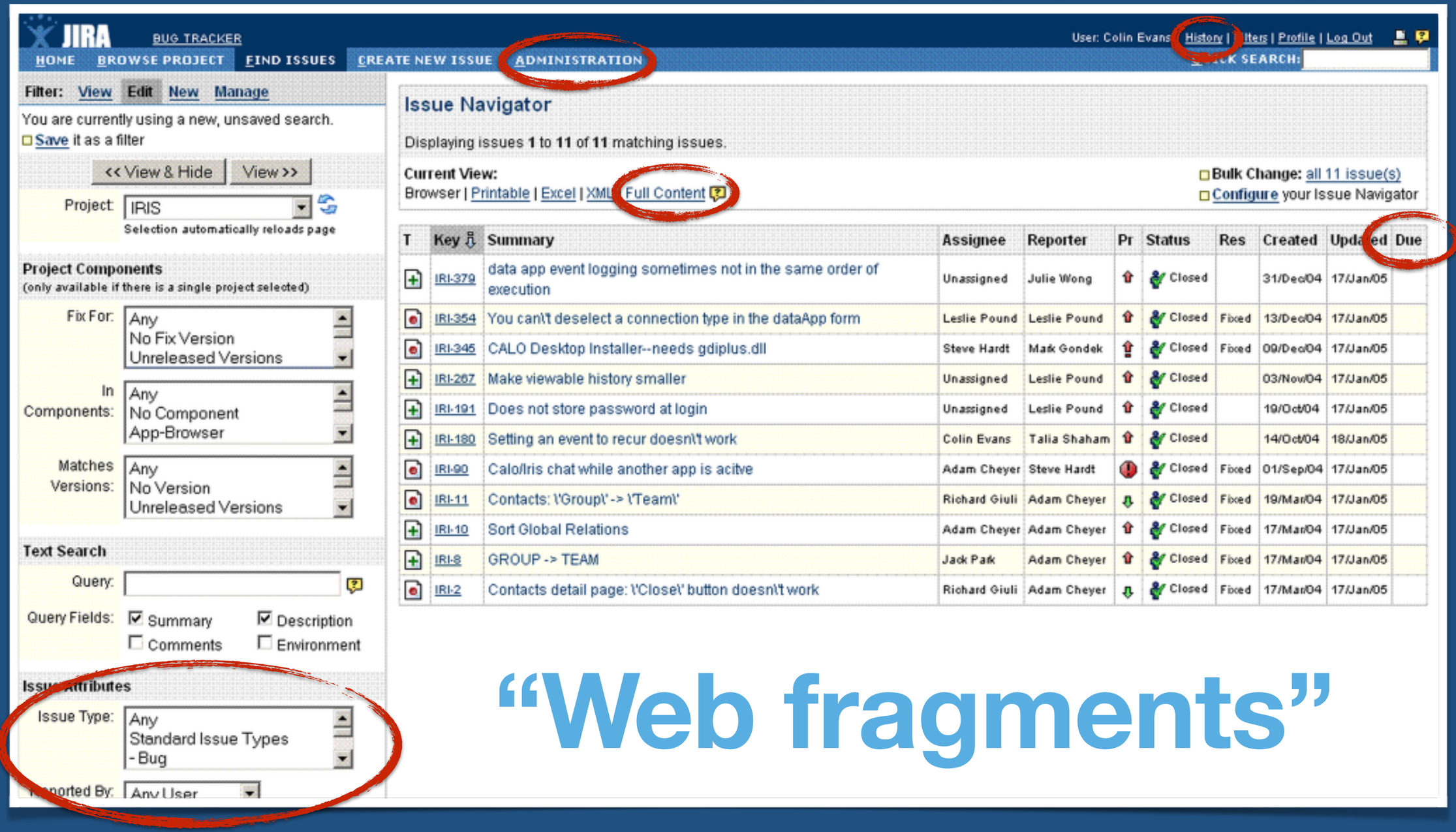
Developer support load

Atlassian Plugins (2004)

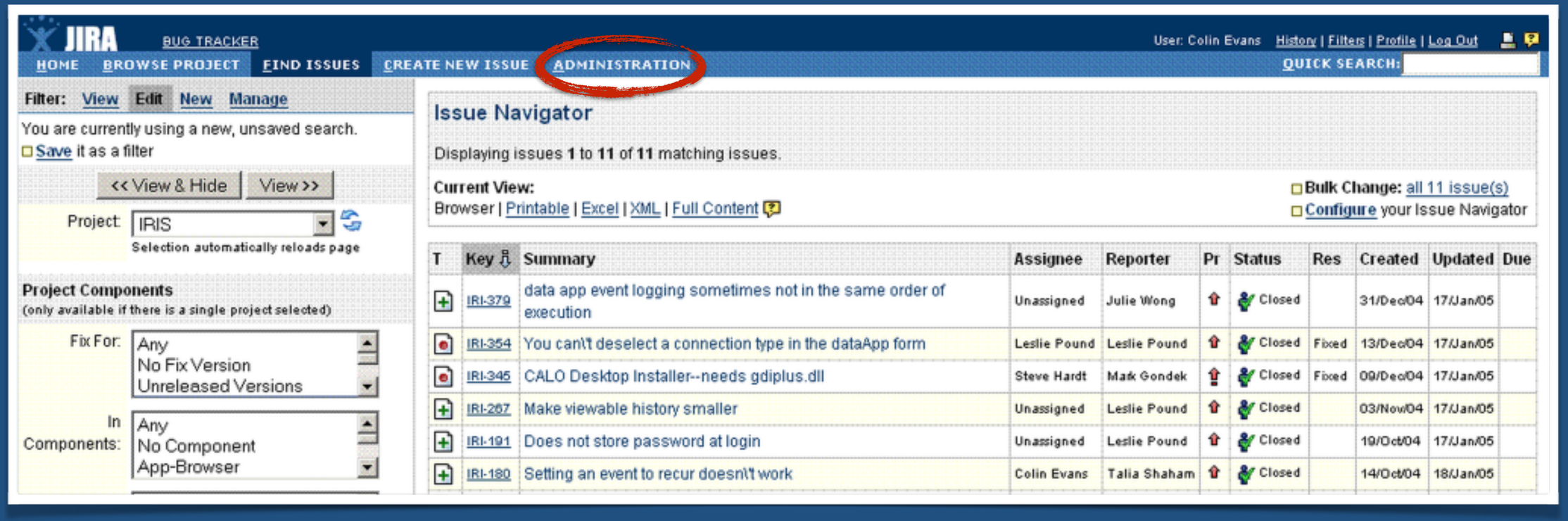

JIRA
BUG TRACKER
User: Colin Evans History | Filters | Profile | Log Out
HOME BROWSE PROJECT FIND ISSUES CREATE NEW ISSUE ADMINISTRATION
QUICK SEARCH:
Filter: View Edit New Manage
You are currently using a new, unsaved search.
Save it as a filter
<< View & Hide View >>
Project: IRIS
Selection automatically reloads page
Project Components
(only available if there is a single project selected)
Fix For: Any No Fix Version Unreleased Versions
In Components: Any No Component App-Browser
Matches Versions: Any No Version Unreleased Versions
Text Search
Query:
Query Fields: Summary Description Comments Environment
Issue Attributes
Issue Type: Any Standard Issue Types - Bug
Reported By: Any User
Issue Navigator
Displaying issues 1 to 11 of 11 matching issues.
Current View:
Browser | Printable | Excel | XML | Full Content
Bulk Change: all 11 issue(s)
Configure your Issue Navigator
T Key Summary Assignee Reporter Pr Status Res Created Updated Due
IRI-379 data app event logging sometimes not in the same order of execution Unassigned Julie Wong Closed 31/Dec/04 17/Jan/05
IRI-354 You can\'t deselect a connection type in the dataApp form Leslie Pound Leslie Pound Closed Fixed 13/Dec/04 17/Jan/05
IRI-345 CALO Desktop Installer--needs gdiplus.dll Steve Hardt Mark Gondek Closed Fixed 09/Dec/04 17/Jan/05
IRI-267 Make viewable history smaller Unassigned Leslie Pound Closed 03/Nov/04 17/Jan/05
IRI-191 Does not store password at login Unassigned Leslie Pound Closed 19/Oct/04 17/Jan/05
IRI-180 Setting an event to recur doesn\'t work Colin Evans Talia Shaham Closed 14/Oct/04 18/Jan/05
IRI-90 Calo/Iris chat while another app is acitve Adam Cheyer Steve Hardt Closed Fixed 01/Sep/04 17/Jan/05
IRI-11 Contacts: \'Group\' -> \'Team\' Richard Giuli Adam Cheyer Closed Fixed 19/Mar/04 17/Jan/05
IRI-10 Sort Global Relations Adam Cheyer Adam Cheyer Closed Fixed 17/Mar/04 17/Jan/05
IRI-8 GROUP -> TEAM Jack Park Adam Cheyer Closed Fixed 17/Mar/04 17/Jan/05
IRI-2 Contacts detail page: \'Close\' button doesn\'t work Richard Giuli Adam Cheyer Closed Fixed 17/Mar/04 17/Jan/05
JIRA 3.0

How?

Wherever there is a collection of type FOO, let the end user inject their own implementations of FOO.



“Web fragments”




```
<ul>
  <li><a href="/dashboard.jspa">Home</a></li>
  <li><a href="/project.jspa">Browse Project</a></li>
  <!-- … -->
  <li><a href="/admin.jspa">Administration</a></li>
</ul>
```

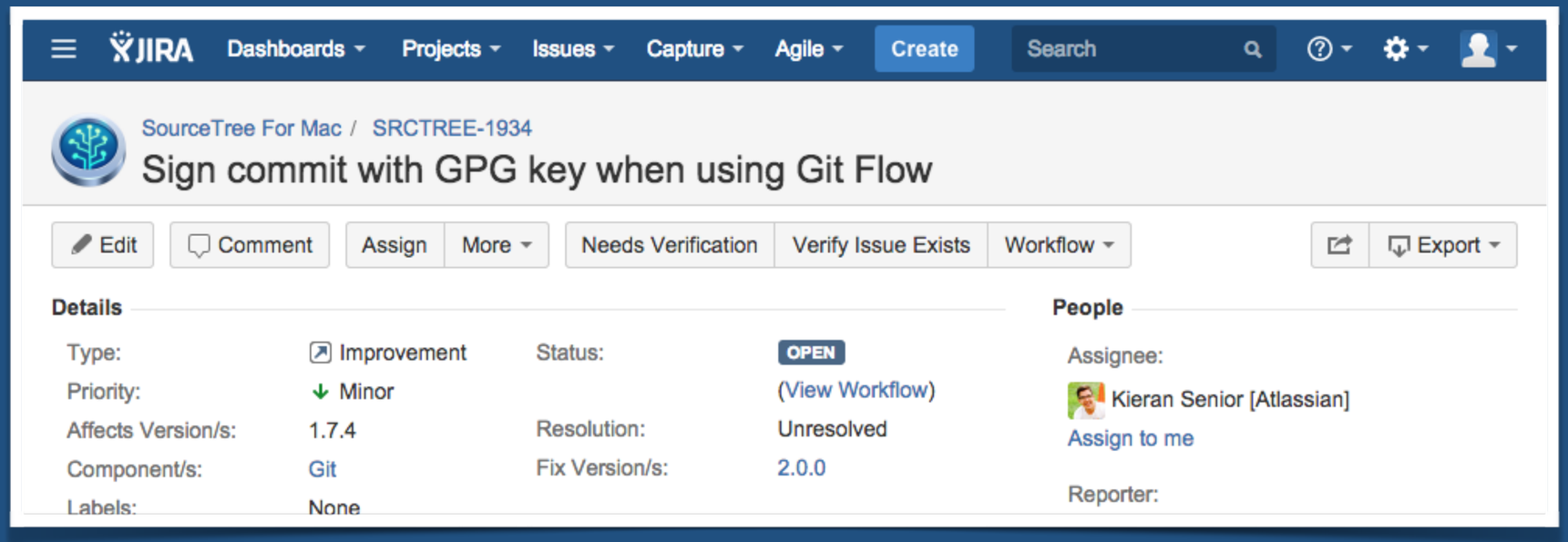
```
interface WebItem {
    String getHref();
    String getText();
}

class FooWebLink implements WebItem {
    public String getHref() {
        return “http://foo.com”;
    }

    public String getText() {
        return “Foo”;
    }
}
```

PluginAccessor

```
List<WebItem> getModules()
```

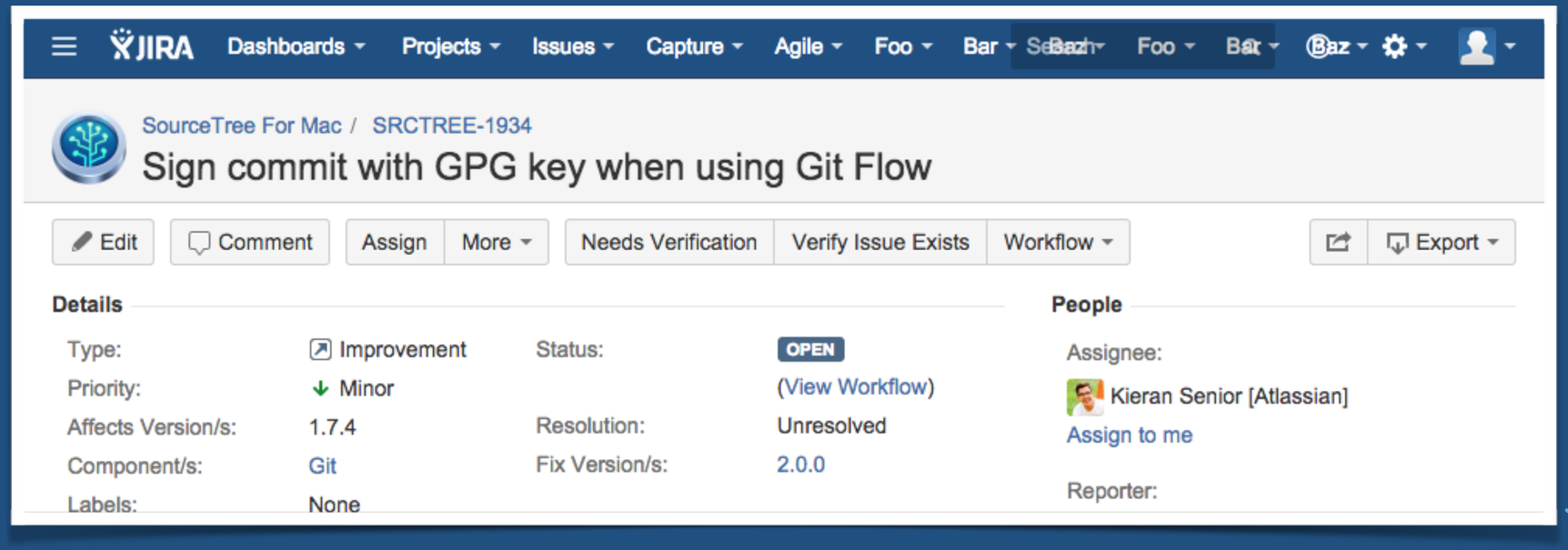
```
<ul>
  <li><a href="/dashboard.jspa">Home</a></li>
  <li><a href="/project.jspa">Browse Project</a></li>
  <!-- … -->
  <li><a href="/admin.jspa">Administration</a></li>
</ul>
  <% for (WebItem wi : pluginAccessor.getModules()) { %>
    <li>
      <a href="<% wi.getHref() %>">
        <% wi.getText() %>
      </a>
    </li>
  <% } %>
```

Plugins promote abstraction and enforce modular architecture!



Two plugins looks great!

3+ plugins.. not so hot.



Every (pluggable) collection is now potentially infinitely large.

PluginAccessor

```
IList<IModule> GetModules(Type)
```

```
public void transitionIssue(Issue issue, Transition tn) {

    // update issue state
    // ..

    // send email to watchers
    for (User watcher : issue.getWatchers()) {
        issue.emailTo(watcher);
    }

}
```

```
public void transitionIssue(Issue issue, Transition tn) {
    // update issue state
    // ..
}

public static interface PostFn {
    void onTransition(Issue issue);
}

public class EmailFn implements PostFn {
    public void onTransition(issue) {
        for (User watcher : issue.getWatchers()) {
            issue.emailTo(watcher);
        }
    }
}
```

```
public void transitionIssue(Issue issue, Transition tn) {
    // update issue state
    // ..
    // trigger post functions
    for (PostFn fn : pluginAccessor.get(PostFn.class)) {
        fn.onTransition(issue);
    }
}

public class EmailFn implements PostFn {
    public void onTransition(issue) {
        for (User watcher : issue.getWatchers()) {
            issue.emailTo(watcher);
        }
    }
}
```

Adding a new “plugin point” is easy if you already have good abstraction!

You SHOULD make “pluginizing” a feature as simple & quick as possible. Otherwise if there’s a crunch for time, it won’t happen.

```
for (PostFn fn : pluginAccessor.get(PostFn.class)) {
    fn.onTransition(issue);
}
```

StackOverflowError

ArithmeticException

OutOfMemoryError

ClassNotFoundException

ClassCastException

while (true) {}

Murphy’s law:
If a plugin can fail, it will.
***Horrifically*.**

Code defensively

```
for (PostFn fn : pluginAccessor.get(PostFn.class)) {
    log.debug("Invoking " + fn);
    try {
        fn.onTransition(issue);
    } catch (Exception e) {
        log.error("Plugin post-function failed", e);
    }
}
```

Corollary:

If an API is public, it will be used.

Often *incorrectly*.

Show them the way

```
/**
 * @param list a non-null, non-empty list
 */
public void publicApiMethod(List<?> list)
  if (list == null) throw new NullPointerException(
    “list must not be null”
  );
  if (list.isEmpty()) throw new IllegalArgumentException(
    “list must not be empty”
  );
  ..
}
```

Backend

- Custom fields
- JQL functions
- Workflows (post-functions, conditions)
- Servlets
- Filters
- Event listeners

generic

Atlassian Plugins

Source (BSD 3-clause licensed)

https://bitbucket.org/atlassian/atlassian-plugins

Embedding Guide

http://bit.ly/embedding-plugins

How it works (1 of 3)

- **Host application exposes some “plugin point” interfaces**

```
public static interface PostFn {
    void onTransition(Issue issue);
}
```

- **Plugin developer provides concrete implementations**

```
public class EmailFn implements PostFn {
    // send email to watchers
    for (User watcher : issue.getWatchers()) {
        issue.emailTo(watcher);
    }
}
```

How it works (2 of 3)

- **Plugin developer adds an XML descriptor (remember it's 2004)**

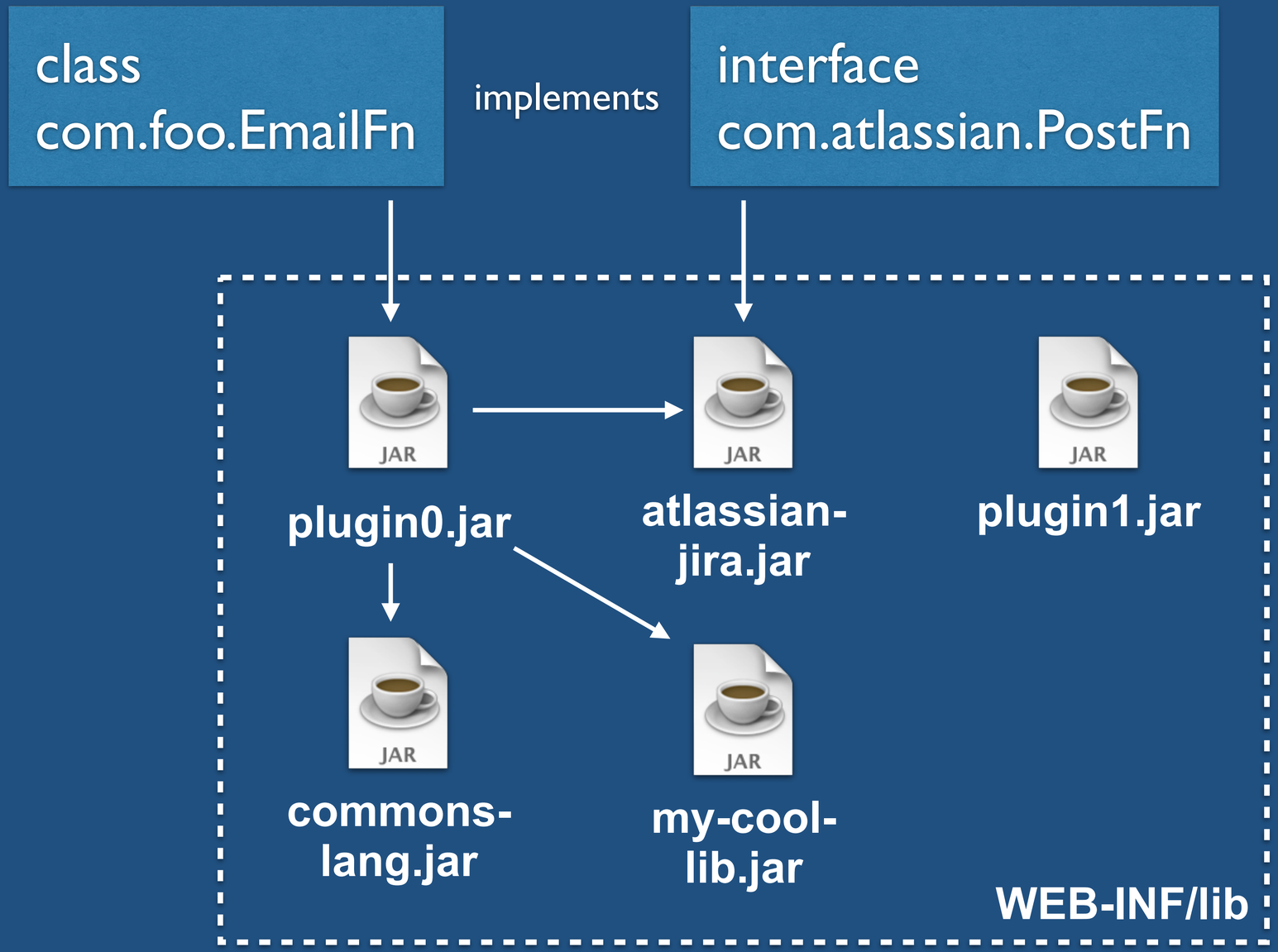
```
<post-function class="com.foo.EmailFn" weight="10" />
```

- **Developer packages up the class and XML in a JAR and puts it on the classpath**

```
atlassian-jira/WEB-INF/lib/my-cool-plugin-1.0.jar/
    atlassian-plugin.xml
    com/foo/EmailFn.class
```

How it works (3 of 3)

- On startup, plugin system parses the XML and instantiates classes with Spring DI
- The `PluginAccessor` is injected into other components and used to look up modules.
- Because the class path is static, the `PluginAccessor` caches plugins very aggressively.


class
com.foo.EmailFn
implements
interface
com.atlassian.PostFn
JAR
plugin0.jar
JAR
atlassian-
jira.jar
JAR
plugin1.jar
JAR
commons-
lang.jar
JAR
my-cool-
lib.jar
WEB-INF/lib

Why it’s awesome!

Easier customization by end-users

Smoother upgrades

Improved customization UX

Architectural benefits

Simple to implement

Why it sucks..

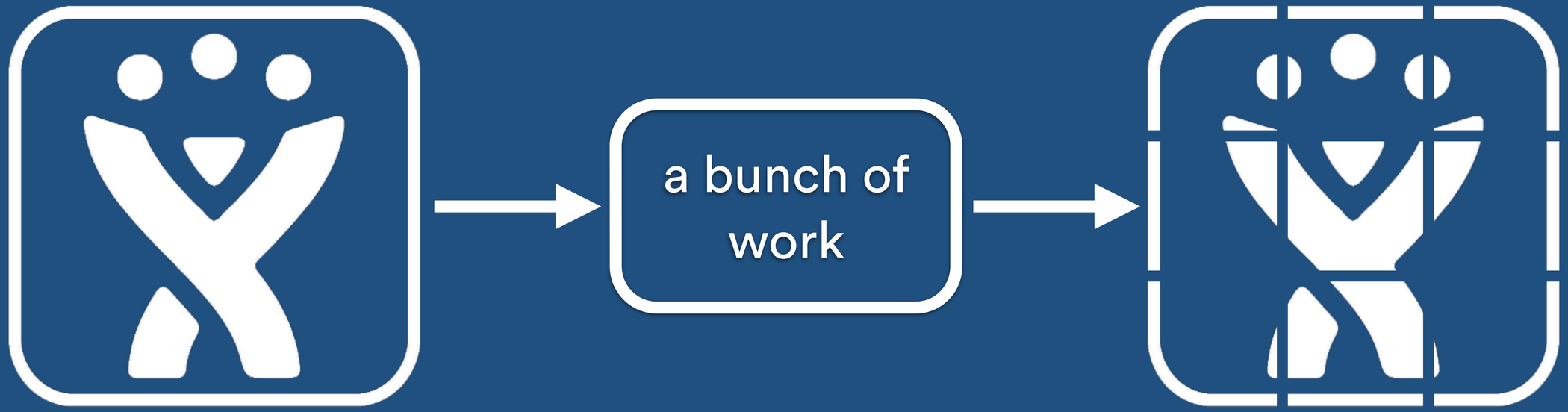
Shared classpath

No API delineation

Need to restart to install plugins

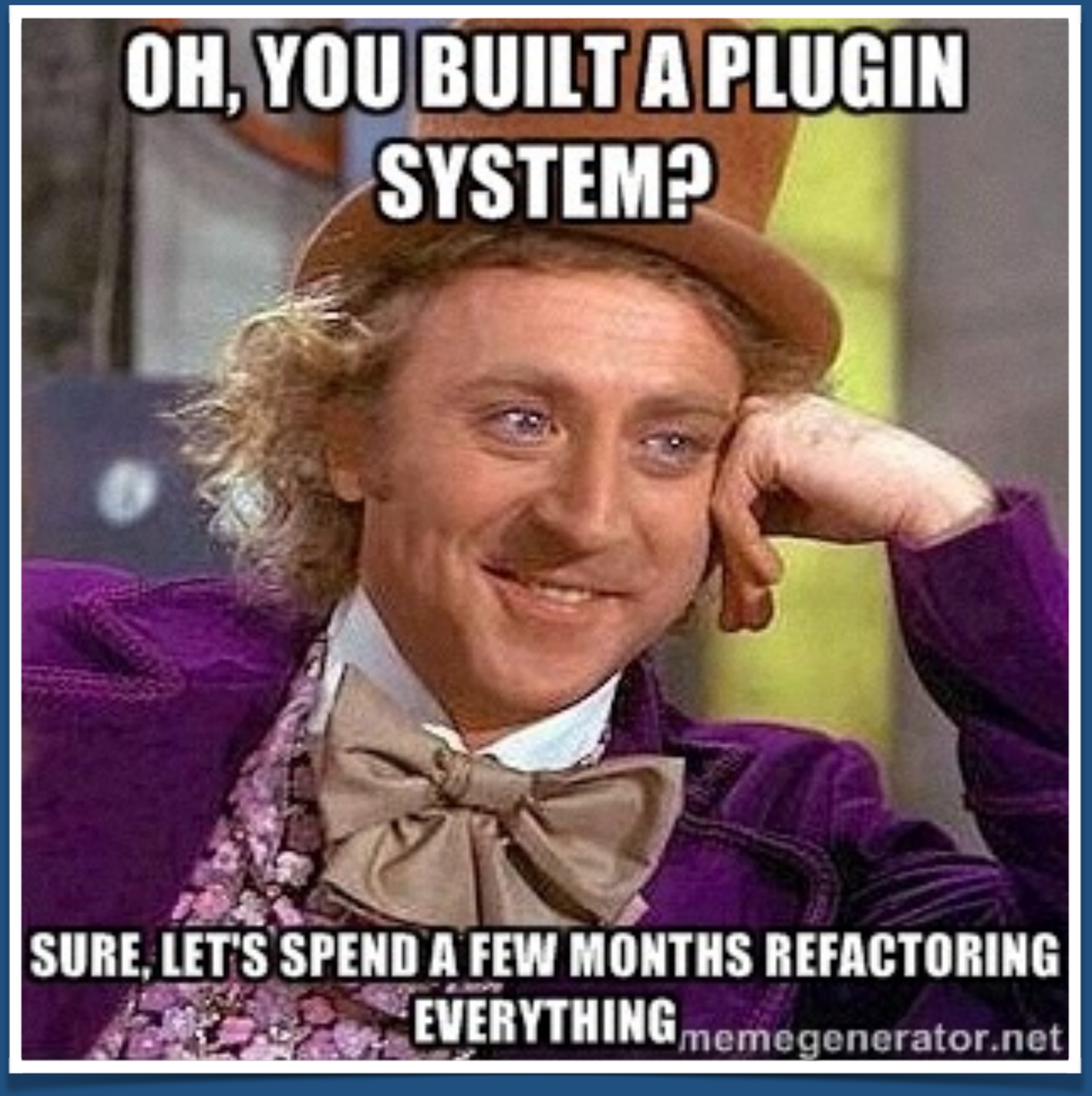
Totally bespoke, non-standard

Re-architecting as plugins



Dogfooding APIs?

product manager



“PM friendly” benefits!

Release plugins out of band

Ship “dark features”

Customers can turn stuff off

Release paid plugins ($$$)

Architecture == velocity

Make sure your plugin system works.

Nothing turns off a plugin developer like a broken API or an unstable platform.

P2
(2008)

P2
(2008)

Why plugins suck..

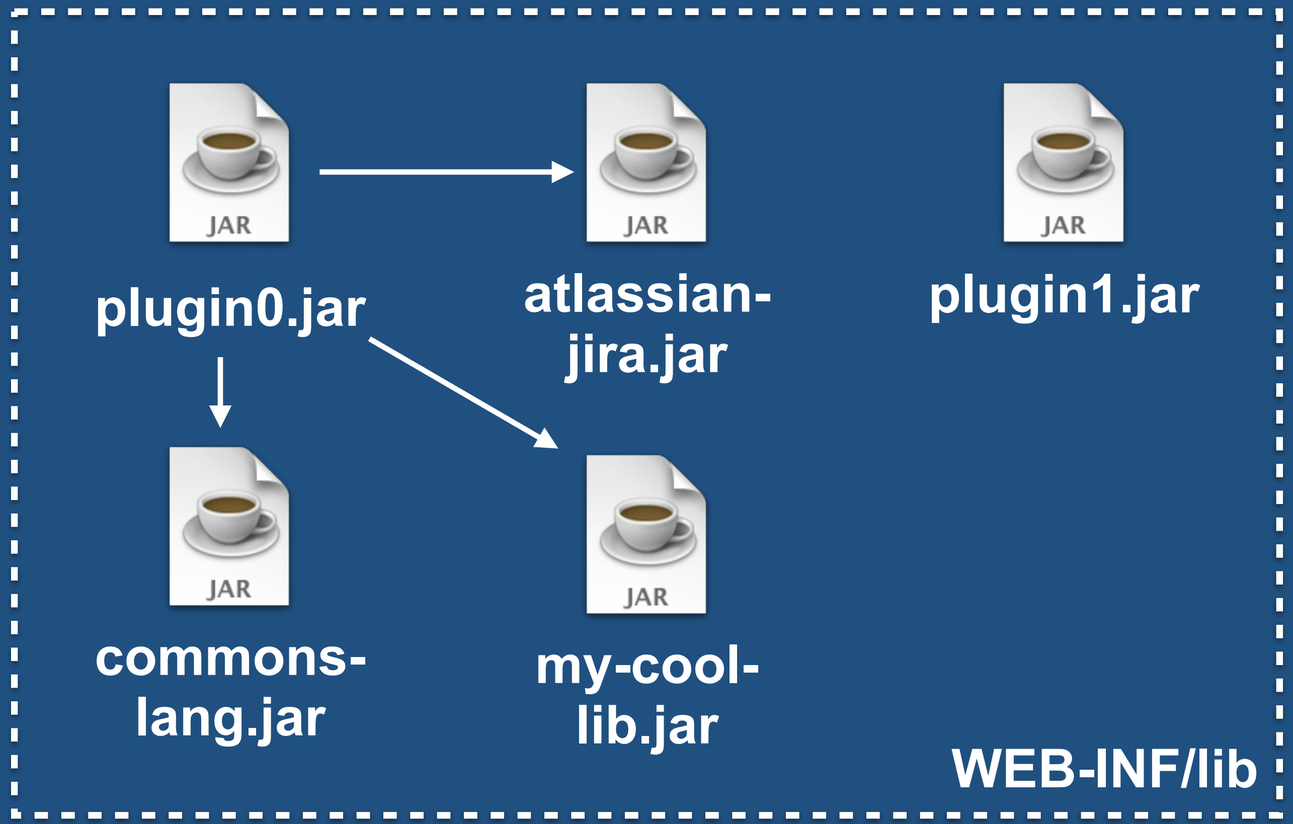
Shared classpath

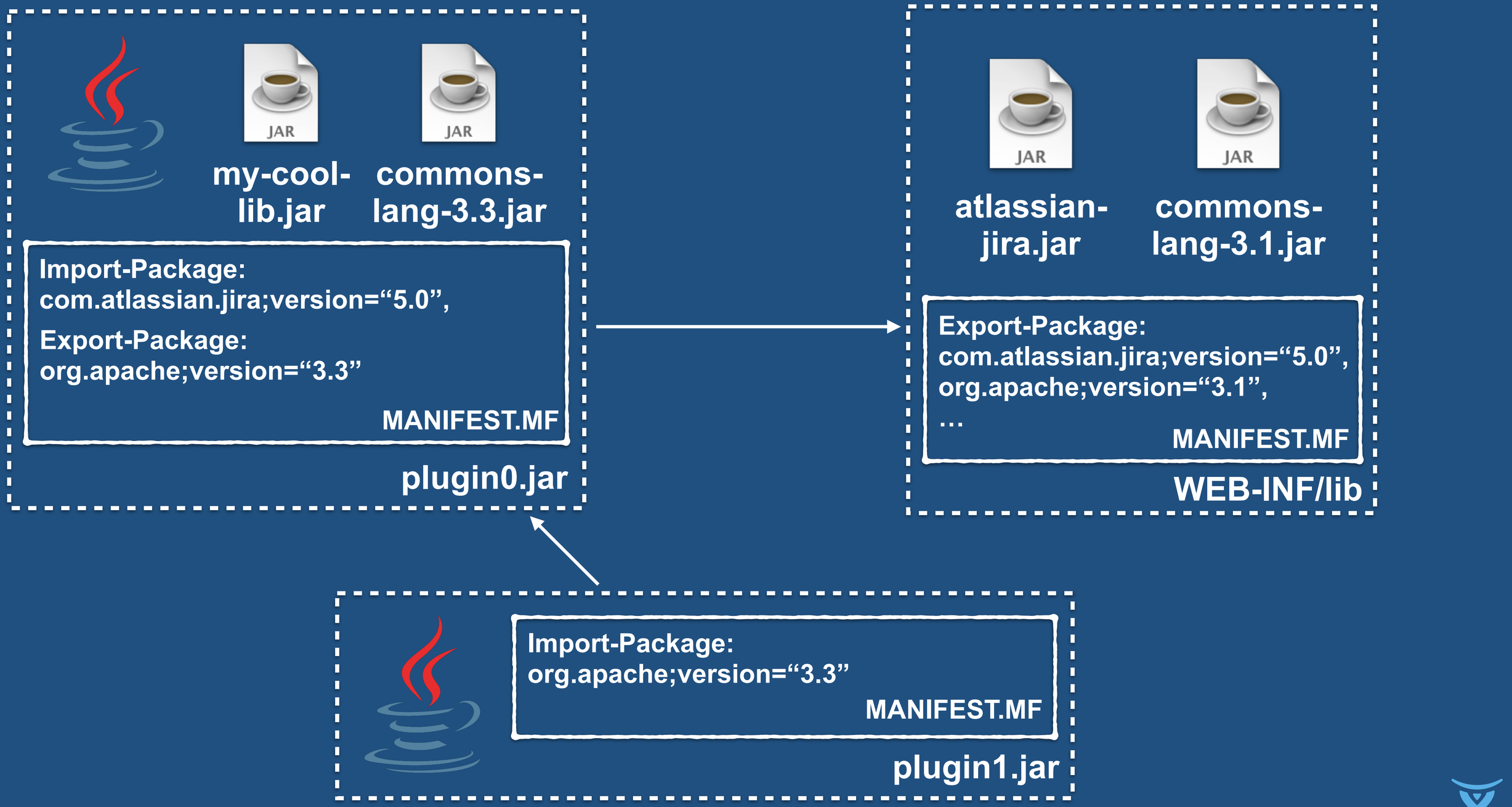
No API delineation

Need to restart to install plugins

Totally bespoke, non-standard

OSGi


JAR
plugin0.jar
JAR
atlassian-jira.jar
JAR
plugin1.jar
JAR
commons-lang.jar
JAR
my-cool-lib.jar
WEB-INF/lib


JAR
JAR
my-cool-lib.jar
commons-lang-3.3.jar
Import-Package:
com.atlassian.jira;version="5.0",
Export-Package:
org.apache;version="3.3"
MANIFEST.MF
plugin0.jar
JAR
JAR
atlassian-jira.jar
commons-lang-3.1.jar
Export-Package:
com.atlassian.jira;version="5.0",
org.apache;version="3.1",
…
MANIFEST.MF
WEB-INF/lib
Import-Package:
org.apache;version="3.3"
MANIFEST.MF
plugin1.jar

Why it sucks..

- ~~Shared classpath~~
- ~~No API delineation~~
- ~~Need to restart to install plugins~~
- ~~Totally bespoke, non-standard~~

Why it sucks..

OSGi is pretty complicated

JIRA Agile's MANIFEST.MF

175Kb of text

P1

```
<post-function class="com.foo.EmailFn"
               weight="10" />
```

+

```
public class EmailFn implements PostFn {..}
```

=

a working plugin

P2

```
<post-function class="com.foo.EmailFn"
               weight="10" />
```

+

```
public class EmailFn implements PostFn {..}
```

+

```
Import-Package:
com.atlassian.jira;version="5.0",
Export-Package:
org.apache;version="3.3"
```

+

```
<component>
  ..
</component>
```

=

a working plugin
… maybe

Abstracting OSGi (and a few other things..)

```
/**
 * Gets the default list of transform stages performed by the transformer. Clients wishing to add stages to the
 * transformation process should use this list as a template rather than creating their own from scratch.
 */
public static ArrayList<TransformStage> getDefaultTransformStages()
{
    return new ArrayList<TransformStage>(Arrays.asList(
            new AddBundleOverridesStage(),
            new ScanInnerJarsStage(),
            new ComponentImportSpringStage(),
            new ComponentSpringStage(),
            new ScanDescriptorForHostClassesStage(),
            new ModuleTypeSpringStage(),
            new HostComponentSpringStage(),
            new GenerateManifestStage()
    ));
}
```

Pros

- Less upgrade pain
- Easy onboarding for new users

Cons

- Startup time
- Leaky abstraction
- Non-standard use of standard tech
- Proxies all the way down
- Complexities

Was it worth it?

Probably.

Photo: Nicholas A. Tonelli
Supporting
your ecosystem

Atlassian Marketplace

Search add-ons...

Find, try, and buy add-ons for Atlassian products

Search add-ons

Search

FILTER ADD-ONS BY PRODUCT

JIRA | CONFLUENCE | DEV TOOLS

Top selling for JIRA

See more »

Good for devs!

Paint-by-numbers marketing

Pricing & licensing APIs

Feedback

Tracking & analytics

Good for customers!

Central authority for add-ons

Less fragmented UX

Ratings & reviews

Unified billing

Focus shifts from user to vendor customization

Building a marketplace is non-trivial

Stash Docs 3.7.1

Developer documentation for extending Stash through plugins or our remote APIs

Highlights

- Creating a Stash plugin
- Stash plugin video tutorial
- Open source Stash plugins...

Plugin types

- Hosting Application Events
- Global Hooks
- Respository Hooks
- Global Merge conditions
- Repository Merge conditions
- Adding commit metadata

REST APIs

- Stash Core REST API
- Audit REST API
- Build Integration REST API
- JIRA Integration REST API
- SSH REST API
- Branch Permissions REST API
- Branch Utils REST API
- Comment Likes REST API

Reference

- API Changes in Stash 3.7 NEW
- Java API
- JavaScript API
- Web UI API
- Plugin Module Types
- Tutorials and Examples

Got Feedback?

Copyright © 2013 Atlassian. All rights reserved · Trademark · Privacy · Last Published: 2015-03-10 · 3.7.1

Tim’s top tip for dev docs

Keep your documentation as close to the code as possible.

Bind the code..

```
 * @response.representation.401.example {@link com.atlassian.stash.rest.data.RestErrors#EXAMPLE}
 * @response.representation.401.doc The currently authenticated user has insufficient permissions to view the
 *                                  specified pull request.
 *
 * @response.representation.404.mediaType application/json
 * @response.representation.404.qname errors
 * @response.representation.404.example {@link com.atlassian.stash.rest.data.RestErrors#EXAMPLE}
 * @response.representation.404.doc The specified repository or pull request does not exist.
 */
@GET
public Response getPullRequests(@Context Repository repository,
                                @QueryParam("direction") @DefaultValue("incoming") String direction,
                                @QueryParam("at") String at,
                                @QueryParam("state") String state,
                                @QueryParam("order") String order,
                                @QueryParam("withAttributes") @DefaultValue("true") Boolean withAttributes,
                                @QueryParam("withProperties") @DefaultValue("true") Boolean withProperties,
                                @Context PageRequest pageRequest) {
    PullRequestDirection prDirection = toDirection(direction);
    PullRequestState prState = toPullRequestState(state);
    PullRequestOrder resultOrder = fromString(order, getDefaultOrderForState(prState));

    return ResponseFactory.ok(
            new RestPage<>(pullRequestService.
```

..to the docs!

GET

/rest/api/1.0/projects/{projectKey}/repos/{repositorySlug}/pull-requests?direction&at&state&order&withAttributes&withProperties

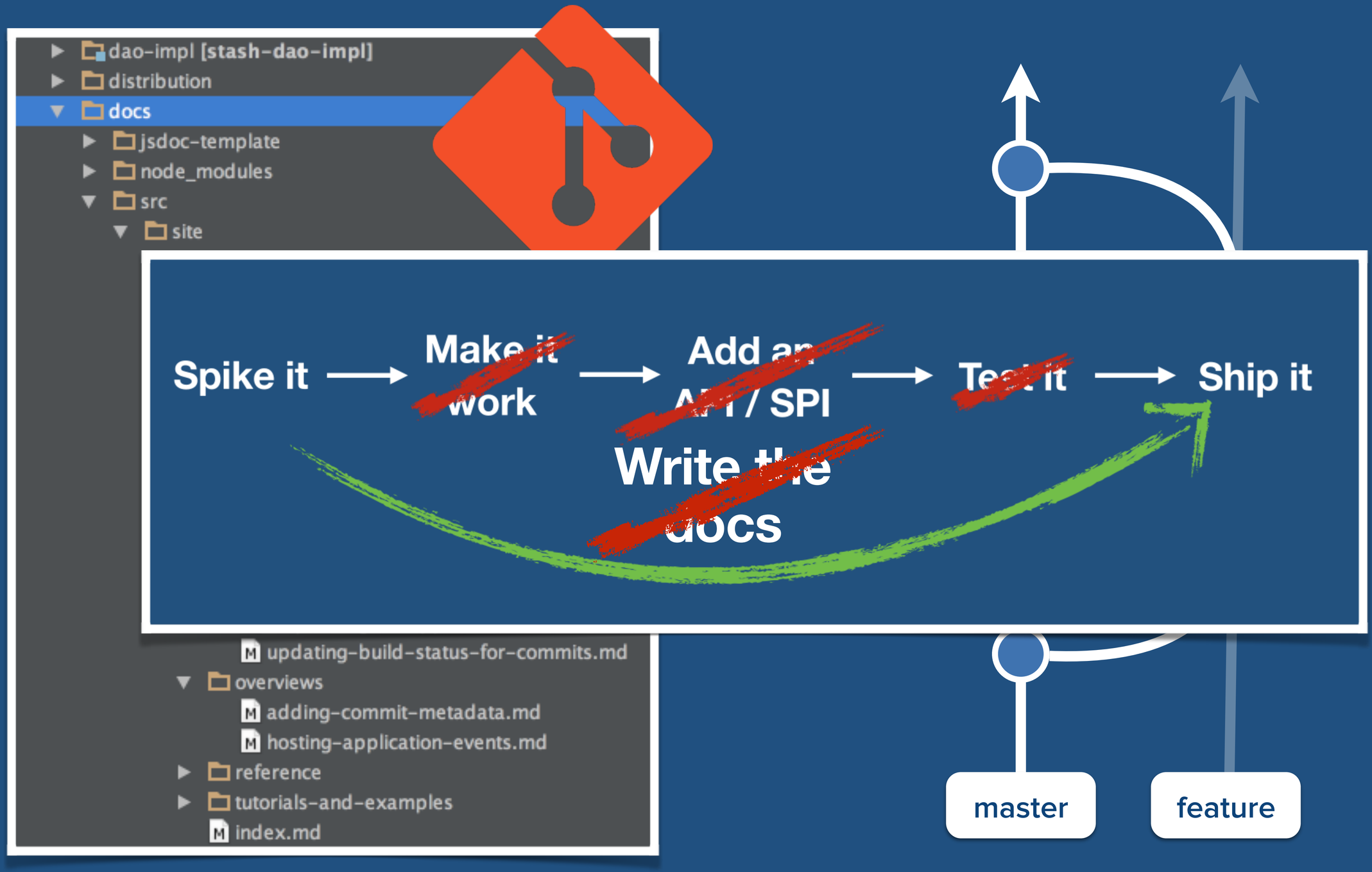
This is a paged API. This API can also be invoked via a user-centric URL when addressing repositories in personal projects.

Retrieve a page of pull requests to or from the specified repository.

The authenticated user must have **REPO_READ** permission for the specified repository to call this resource.

request query parameters

parameter	value	description
direction	*string* Default: `incoming`	(optional, defaults to **INCOMING**) the direction relative to the specified repository. Either **INCOMING** or **OUTGOING**.
at	*string*	(optional) a *fully-qualified* branch ID to find pull requests to or from, such as {@code refs/heads/master}
state	*string*	(optional, defaults to **OPEN**). Supply **ALL** to return pull request in any state. If a state is supplied only pull requests in the specified state will be returned. Either **OPEN**, **DECLINED** or **MERGED**.
order	*string*	(optional) the order to return pull requests in, either **OLDEST** (as in: "oldest first") or **NEWEST**.
withAttributes	*boolean* Default: `true`	(optional) defaults to true, whether to return additional pull request attributes
withProperties	*boolean* Default: `true`	(optional) defaults to true, whether to return additional pull request properties


dao-impl [stash-dao-impl]
distribution
docs
jsdoc-template
node_modules
src
site
Spike it
Make it work
Add an API / SPI
Test it
Ship it
Write the docs
updating-build-status-for-commits.md
overviews
adding-commit-metadata.md
hosting-application-events.md
reference
tutorials-and-examples
index.md
master
feature

Code talks!

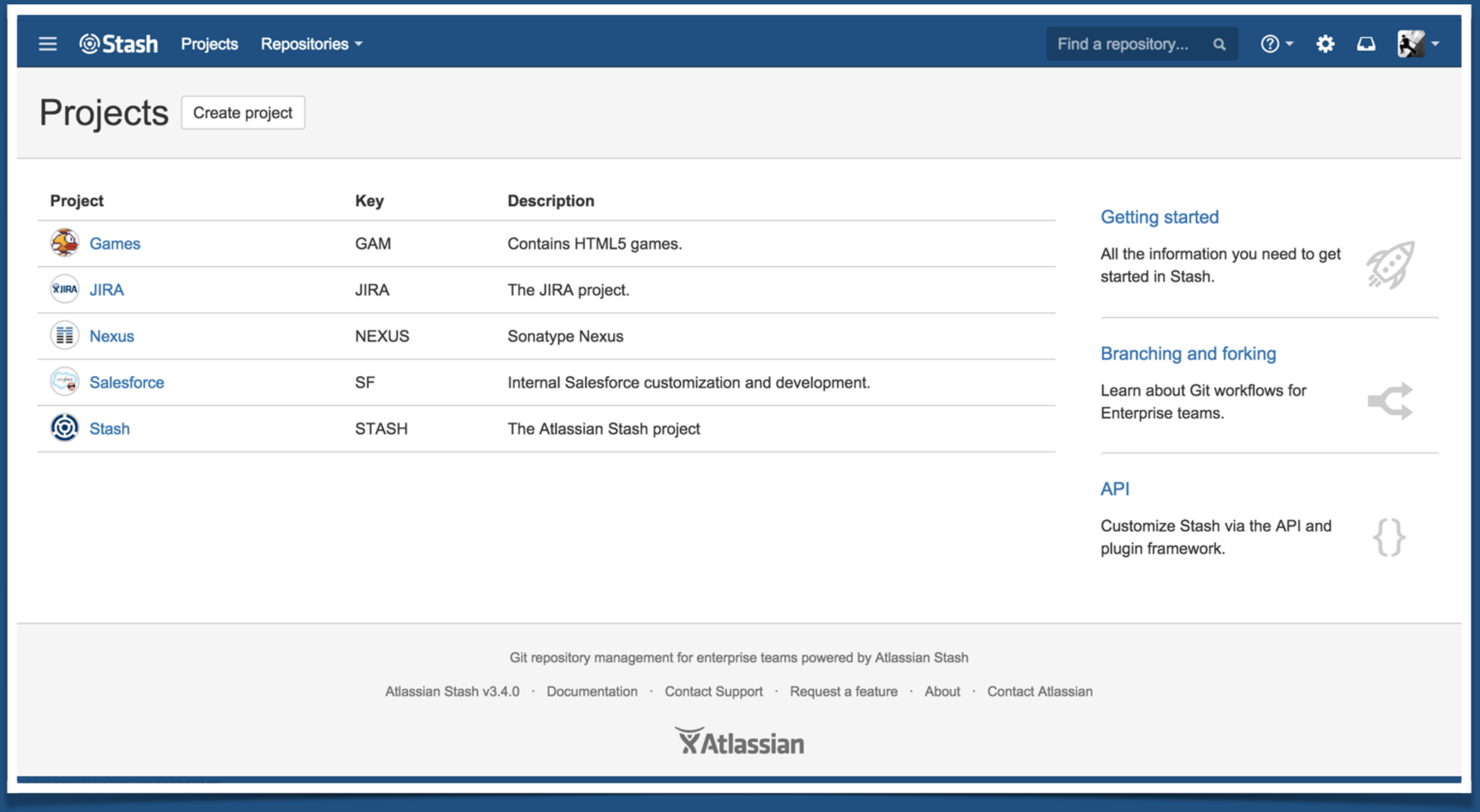
Make sure you have a simple process for open sourcing *and* liberally licensing example add-ons.

Photo: Biser Todorov
Meta-Developer Tools

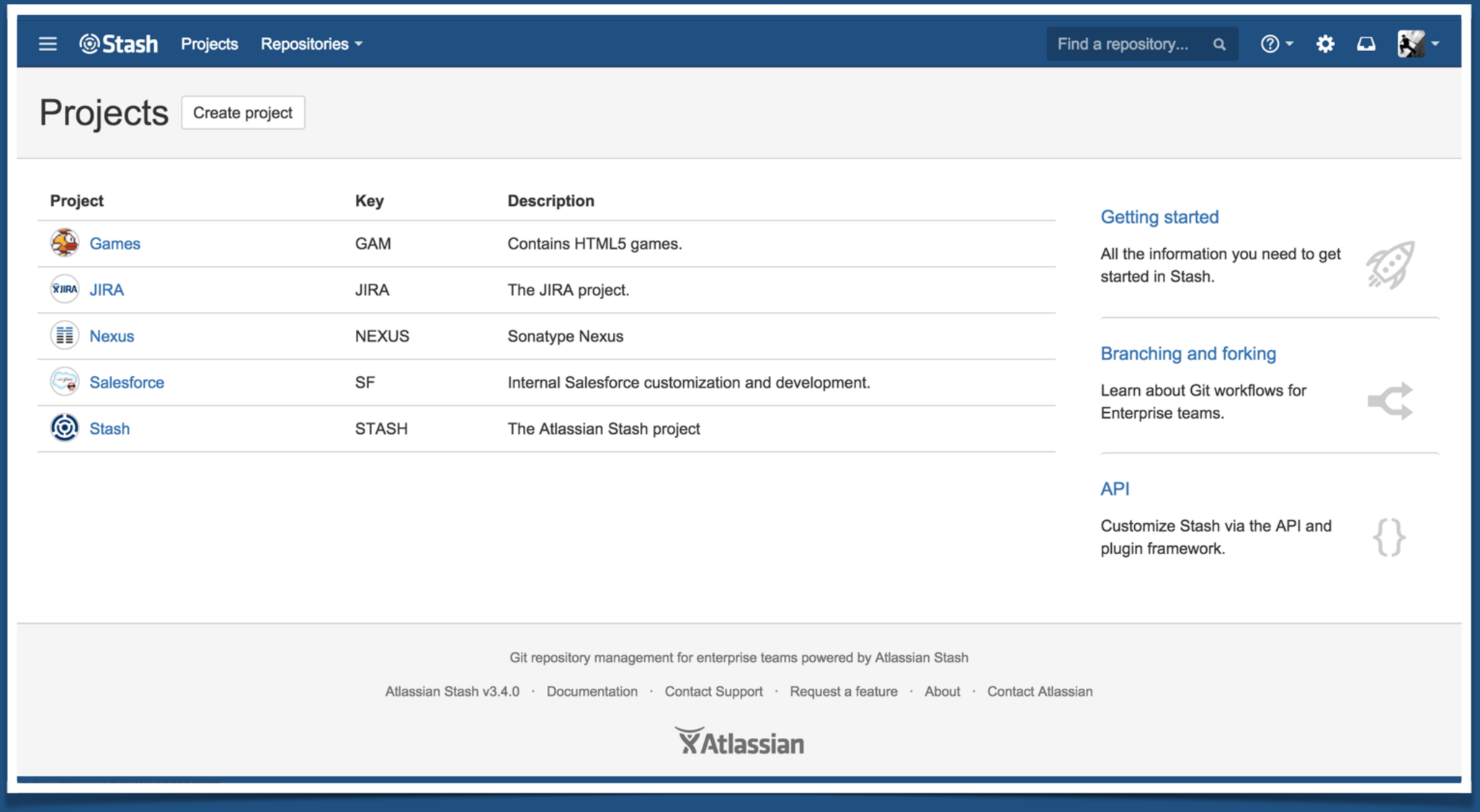
SDK

- Fully automate plugin ALM
- Spin up products in “developer mode”
- Tight development loop
- Provide realistic test data
- Dogfood it!

/stash/projects



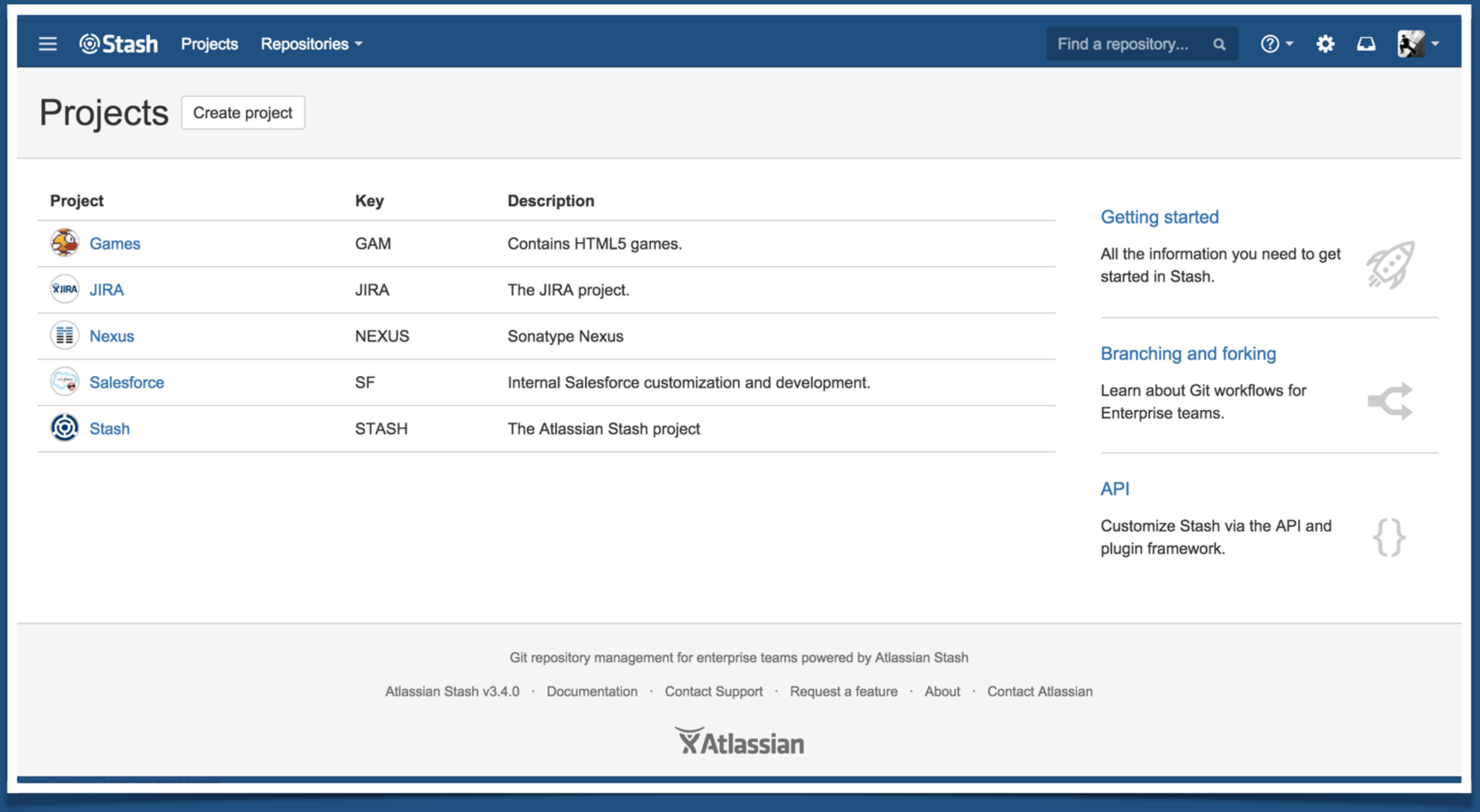
/stash/rest/api/1.0/projects



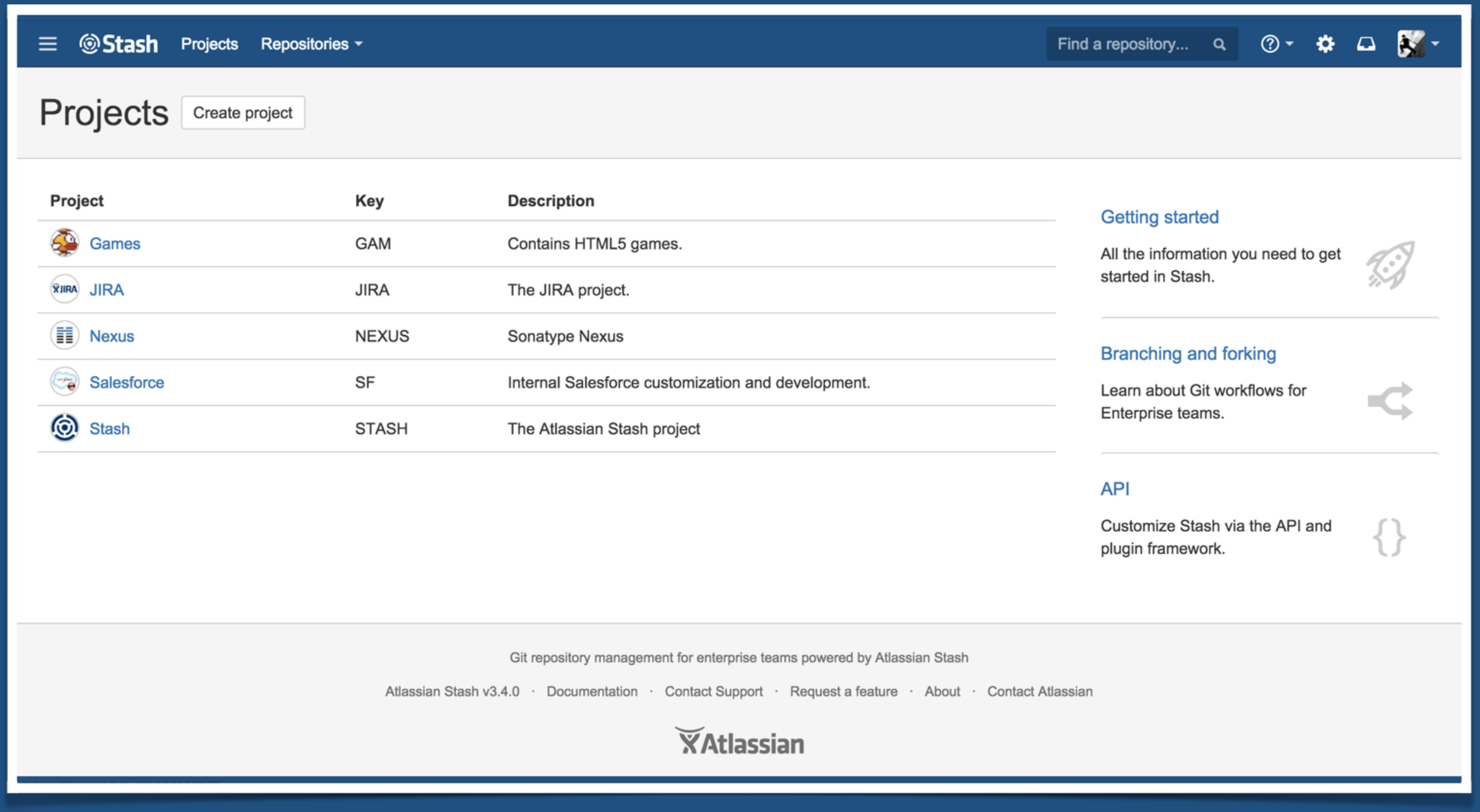
/stash/rest/api/1.0/projects


```
{
    size: 25,
    limit: 25,
    isLastPage: false,
  - values: [
      - {
            key: "ACE",
            id: 1383,
            name: "ACE",
            description: "Atlassian Communications Engine",
            public: false,
            type: "NORMAL",
          - link: {
                url: "/projects/ACE",
                rel: "self"
            },
          - links: {
              - self: [
                  - {
                        href: "https://stash.atlassian.com/projects/ACE"
                    }
                ]
            }
        },
      - {
            key: "ATLASSIAN",
            id: 22,
            name: "atlassian",
            description: "Location for repositories being mirrored from bitbucket's atlassian account. Nothin in here should ever be pushed to!",
            public: false,
            type: "NORMAL",
          - link: {
                url: "/projects/ATLASSIAN",
                rel: "self"
            },
```

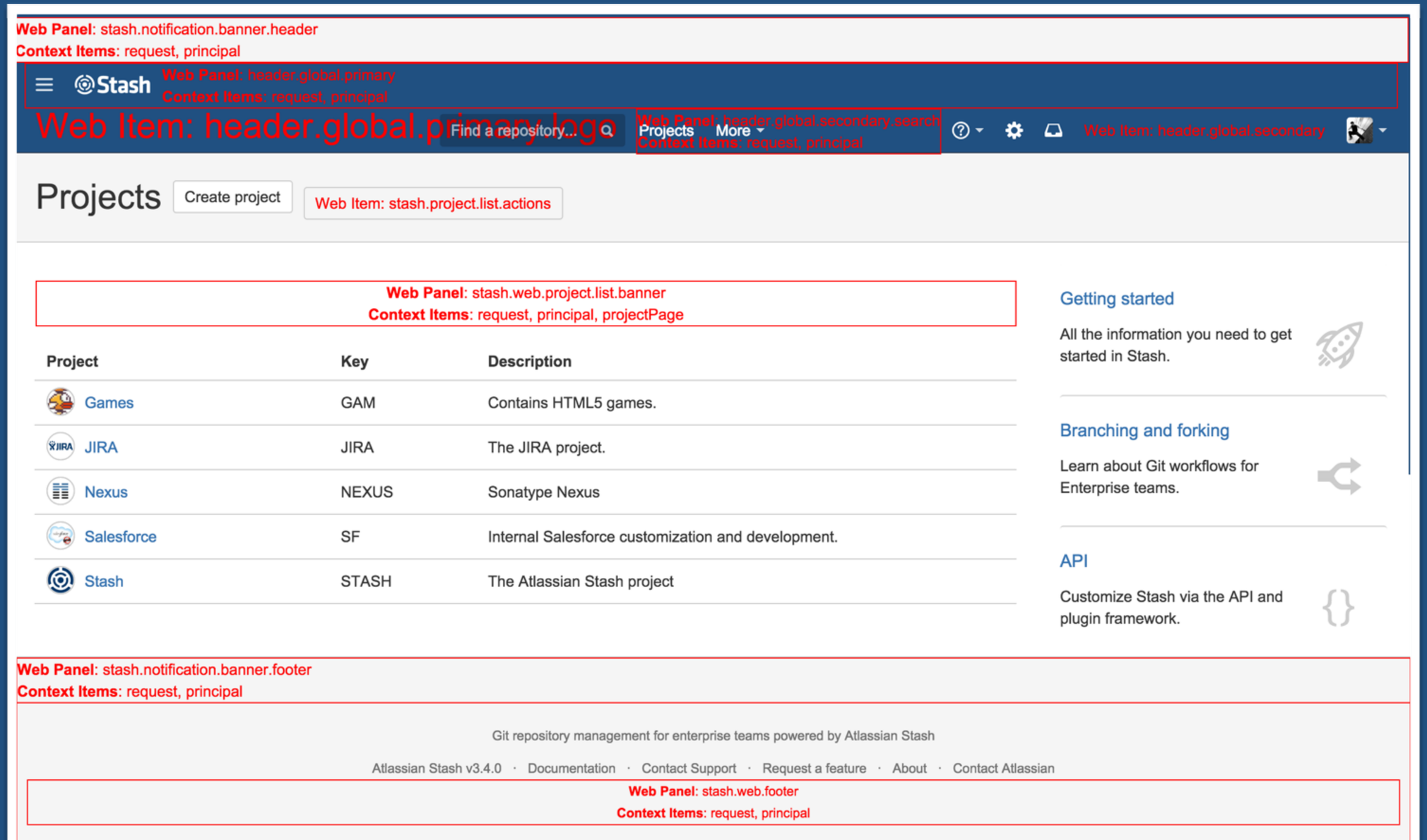
/stash/projects



/stash/projects?web.panels&web.items



/stash/projects?web.panels&web.items



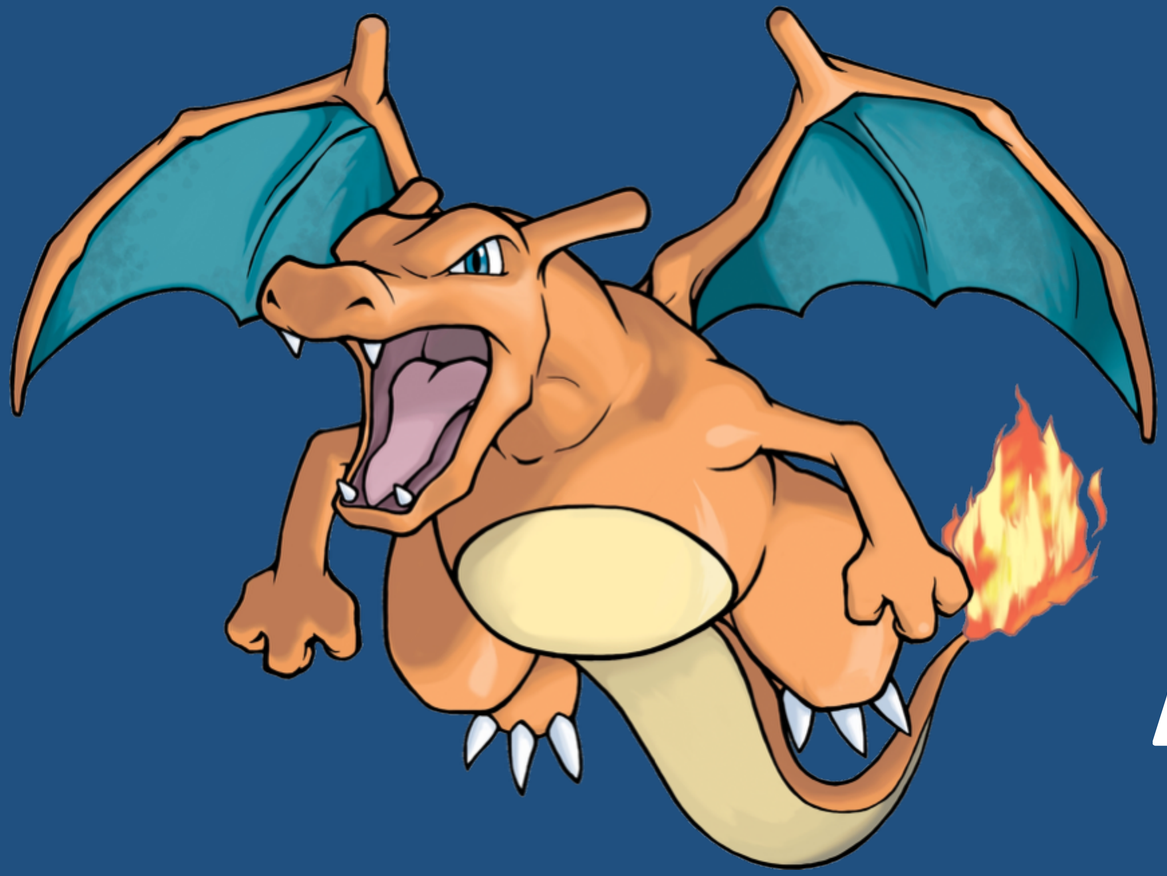
Developer Mode

Trim out dev speed killers (websudo, etc)

Expose logs & debug info

Add in custom “dev mode” plugins

Release your own *internal* productivity tools

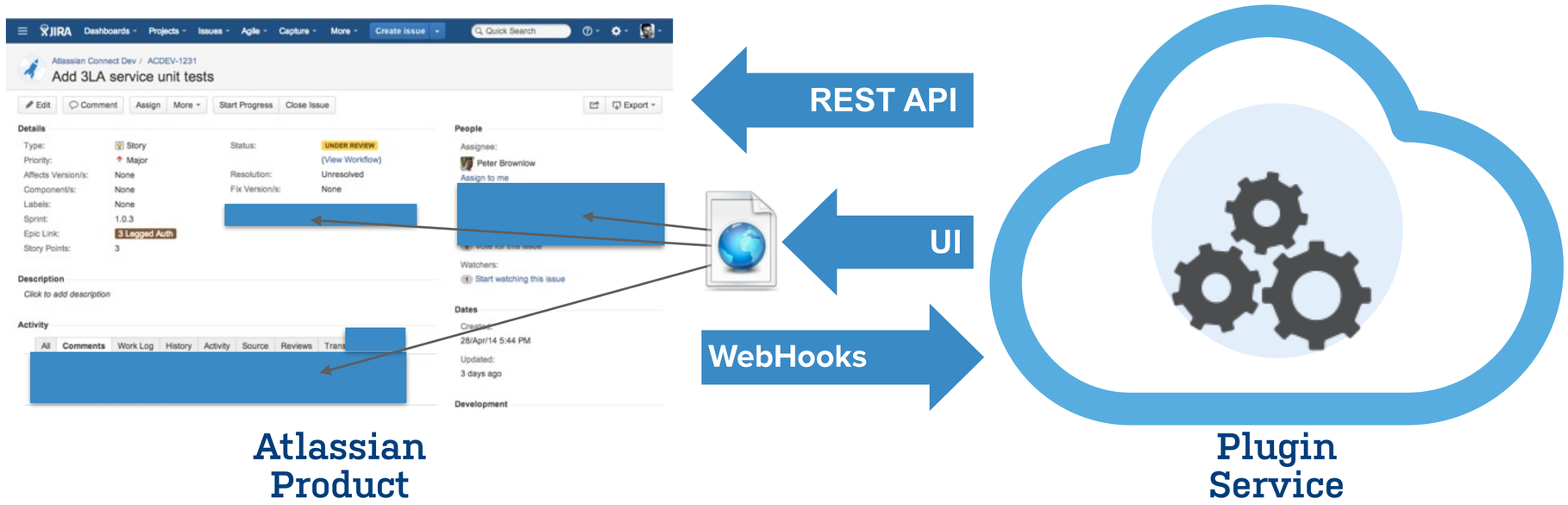


Atlassian Connect (2014)

What is Atlassian Connect?

Remote ~~Plugins~~ Add-ons

- Drive the product via its REST API
- Receive notifications from the product via WebHooks
- Inject content into the product UI via iframes



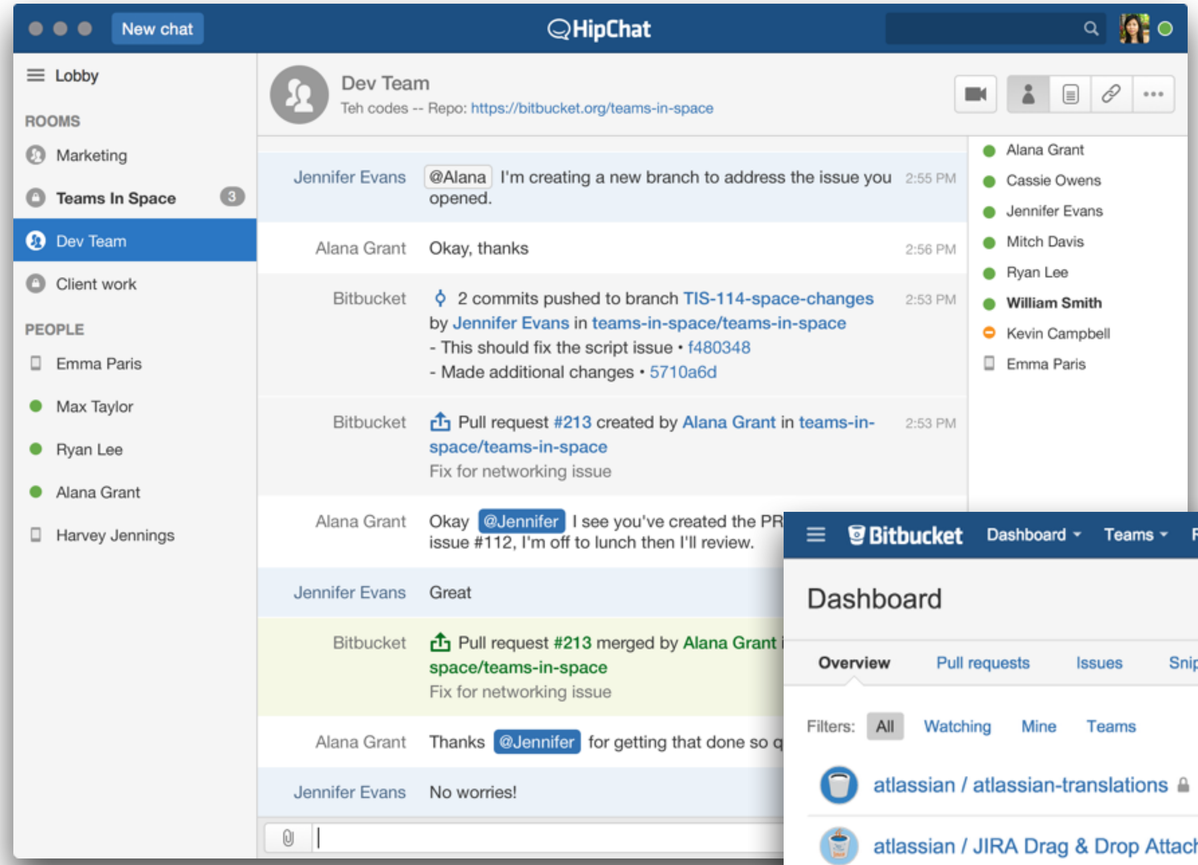
Why?

Atlassian Cloud (SaaS)

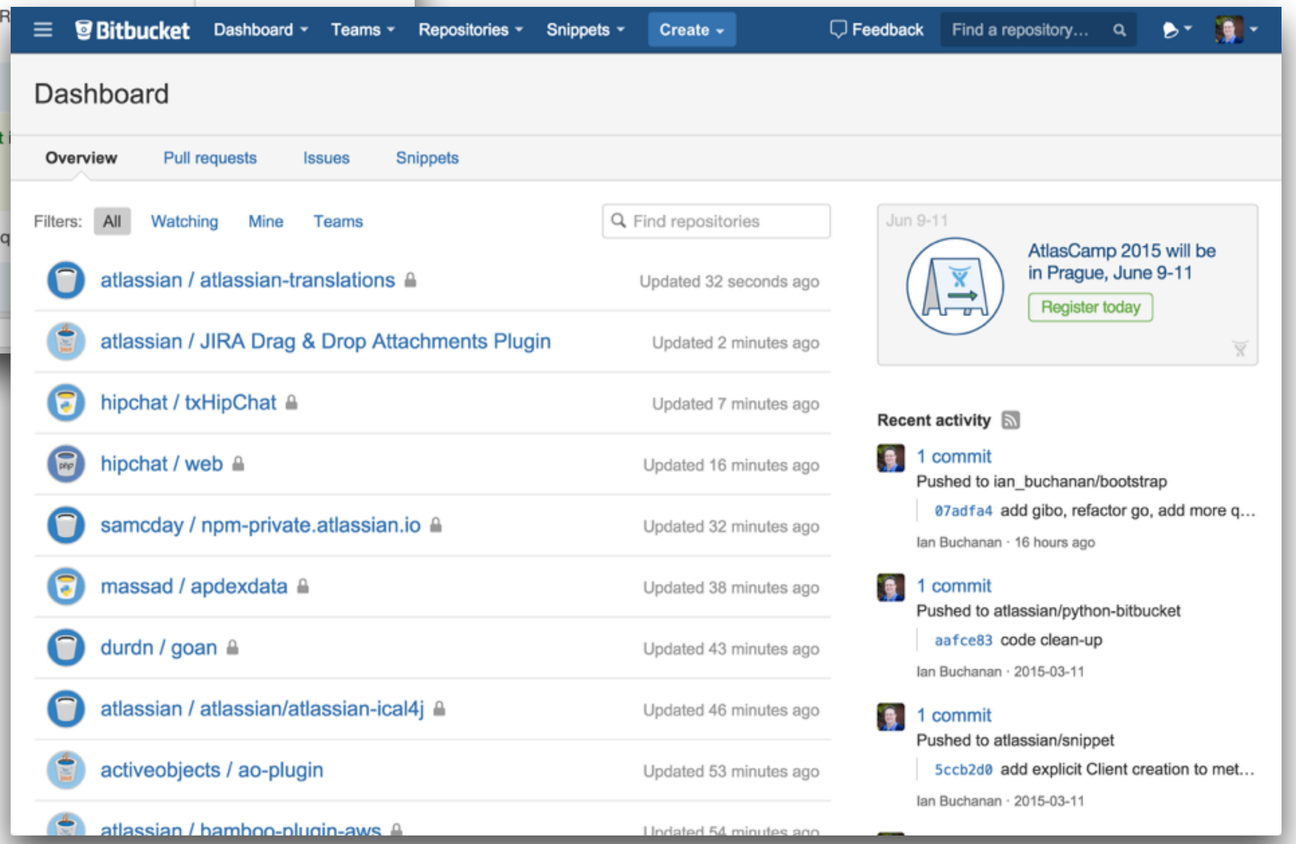
Run 3rd party code out-of-process

Improved customization UX (yet again)

Architectural benefits



HipChat



Bitbucket

Building up

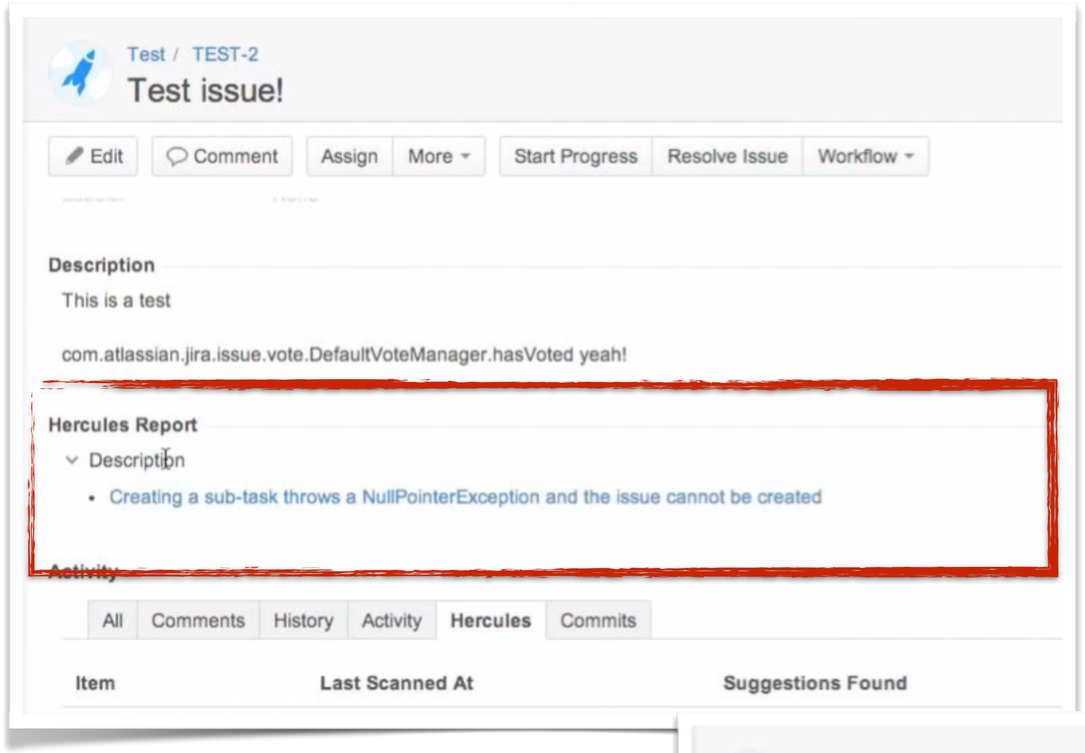
- Connect is a new layer on top of the older plugin model.
- Same Marketplace serves up Connect plugins.
- Same plugins still available in the Marketplace.
- Same REST APIs still available for both Server and Cloud.
- Connect adds a new mode of integration.

http://housebeauty.net/dressy-modern-townhouse-design-with-rooftop-garden-and-glass-walls/eye-catching-green-roofed-penthouse-apartment-rooftop-garden-with-deck-surrounded-by-glass-balustrades/#image-1

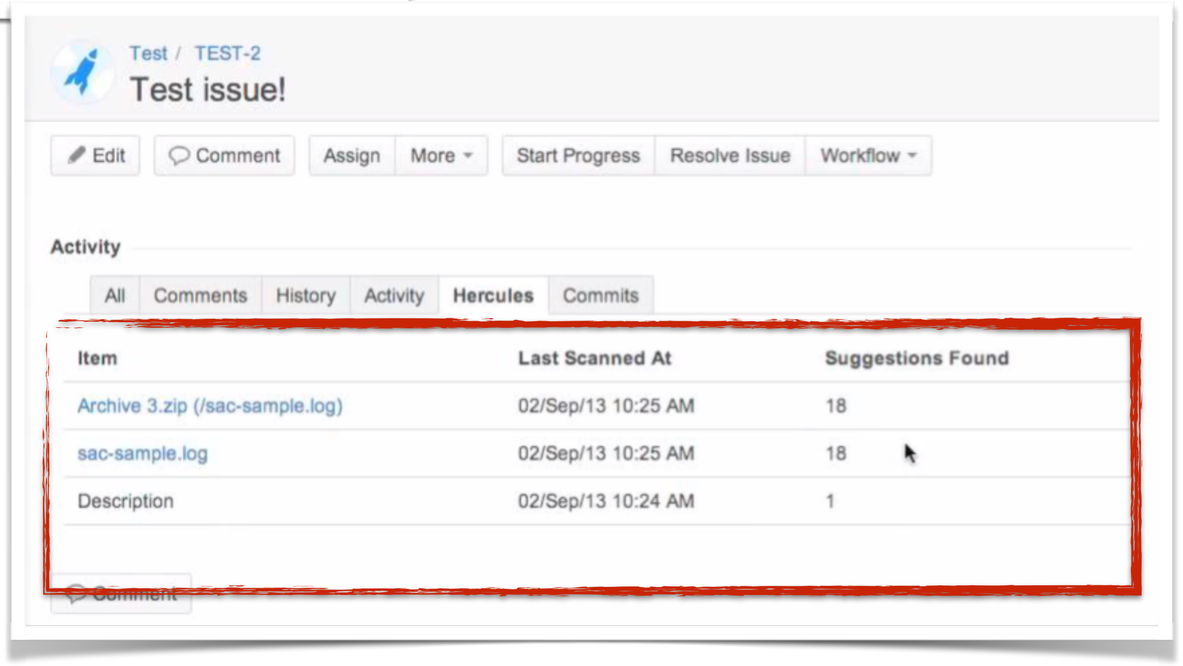
How?

remember web fragments

Wherever a module is called,
let a third-party inject a new feature.



AC WebPanel



AC IssueTab

How to author a Connect Plugin

- Declare plugin with a descriptor.
- Extend the Atlassian application UI with modules.
- Request appropriate scopes.
- Recognize the user.
- Call the Atlassian application's REST API.
- Respond to the Atlassian application's webhooks.
- Register on the Atlassian Marketplace.
- Respect plugin licensing.

```
"name": "Hello World", "description": "…",
"key": "com.example.myaddon", "apiVersion": 1,
"baseUrl": "http://example.com/myaddon",
"modules": {
  "generalPages": [{
    "name": { "value": "Greeting" }, "key": "hello-world",
    "location": "system.top.navigation.bar",
    "url": "/helloworld.html"
  }]
},
"scopes": [ "read", "write" ],
"authentication": { "type": "jwt" },
"lifecycle": { "installed": "/add-on-installed-callback" }
```

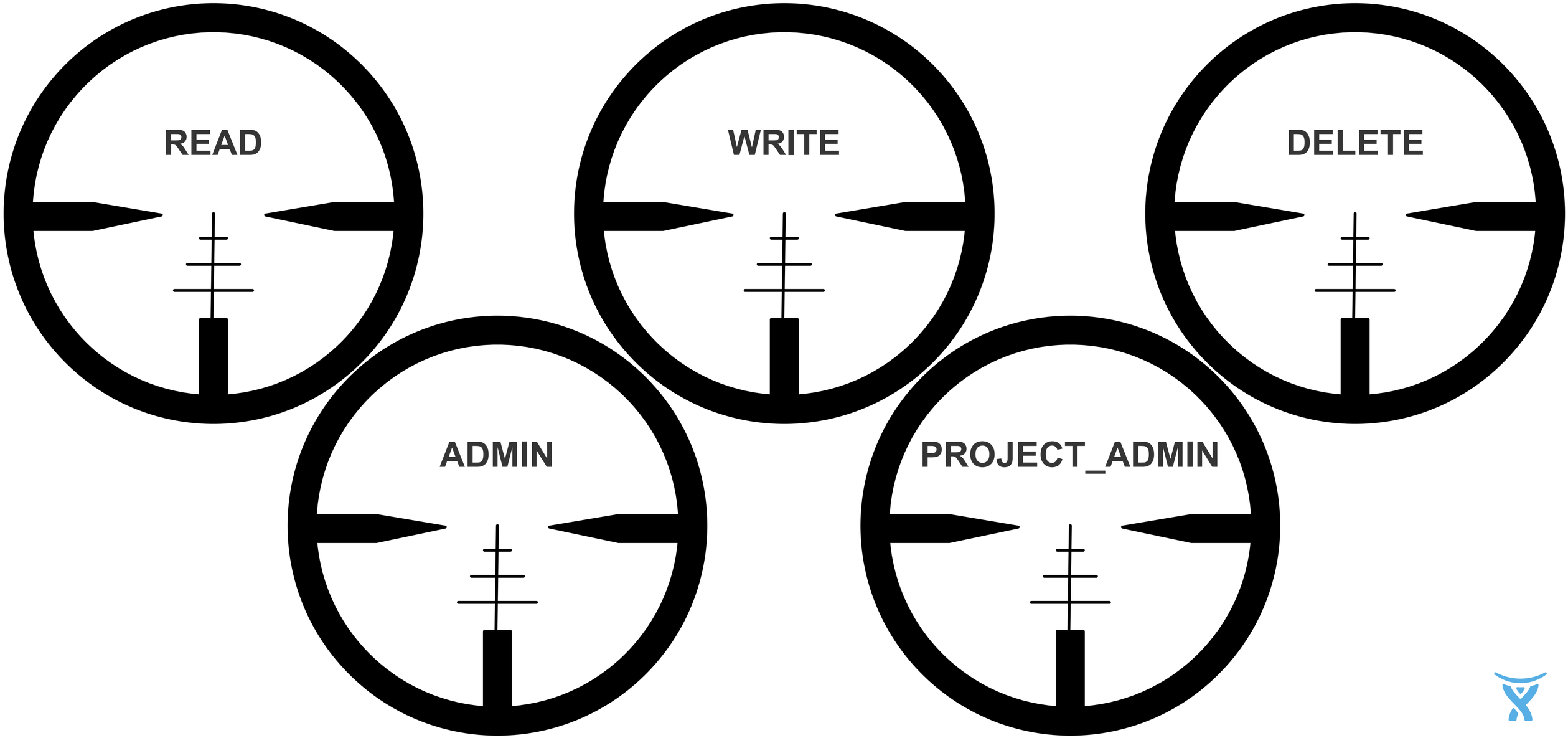
← where it is hosted

← which modules it uses

← which scopes it needs

← recognize the user

Example Scopes for JIRA



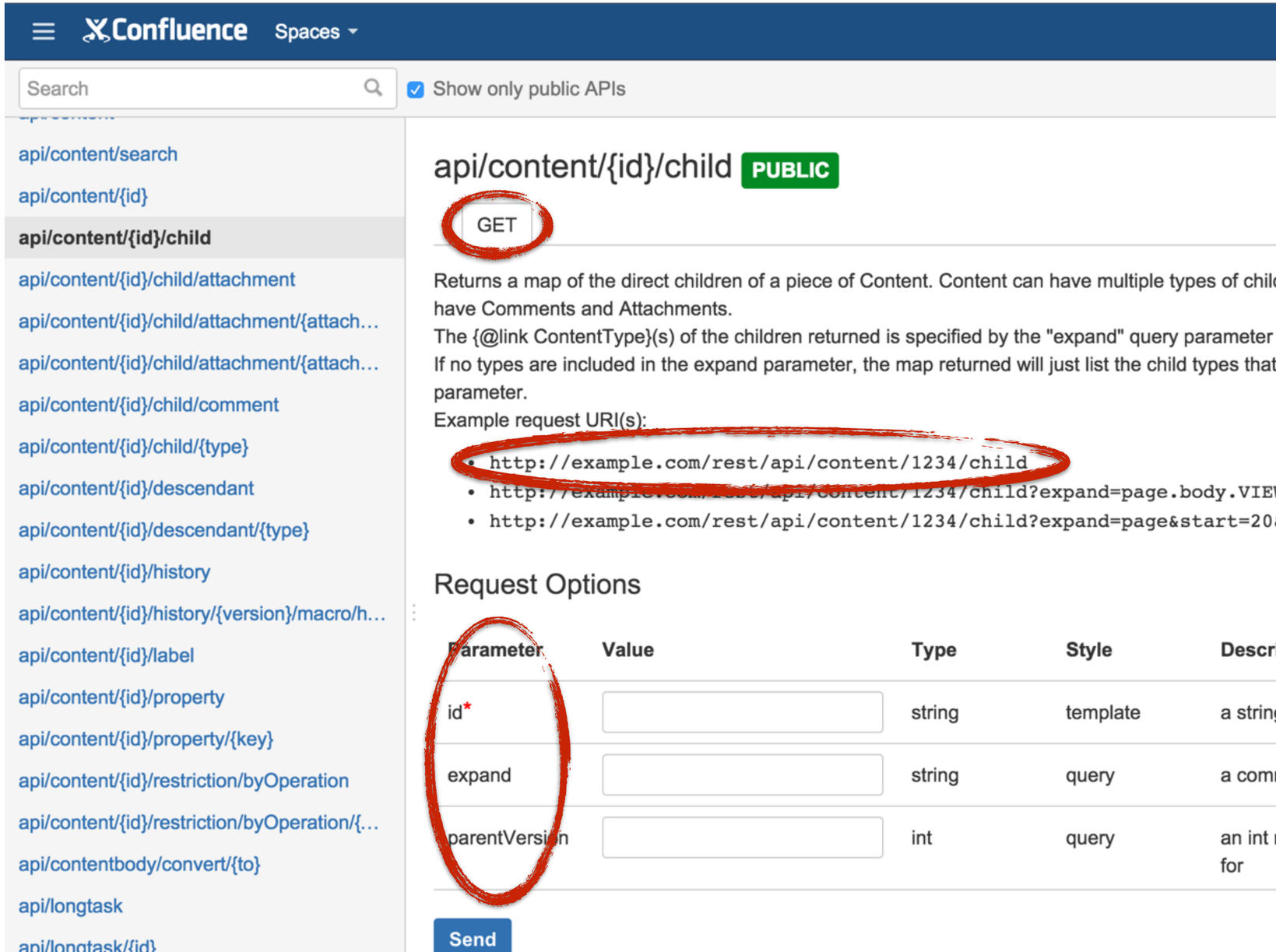
Recognizing the user with JWT

```
GET /myaddon?jwt=<jwt-token> http/1.1
```

```
POST /jira/rest/api/2/issue/AC-1/attachments http/1.1
Authorization: JWT <jwt-token>
```

Example REST API for Confluence

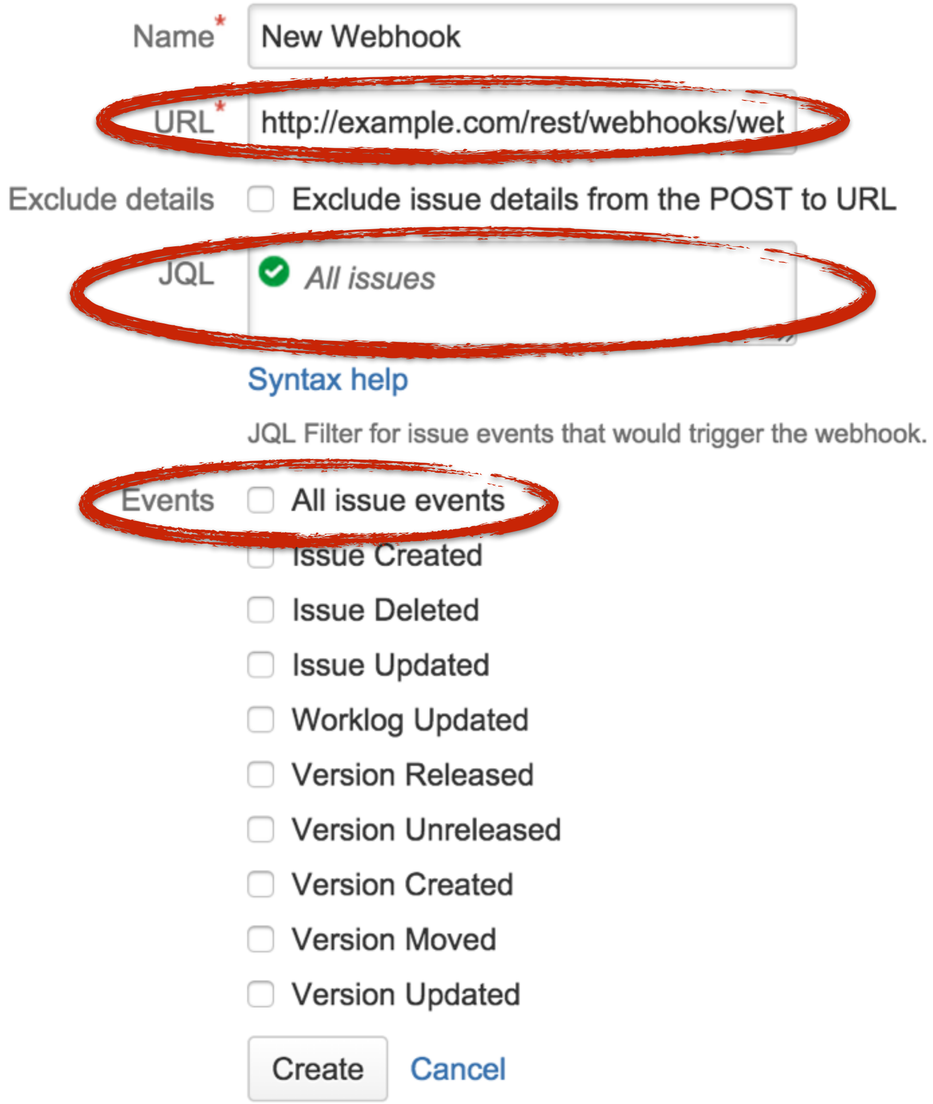
- URLs for resources
- HTTP verbs
- Parmeters filter collections
- JSON data
- HAL(ish) links (newer APIs)



Example JIRA WebHooks

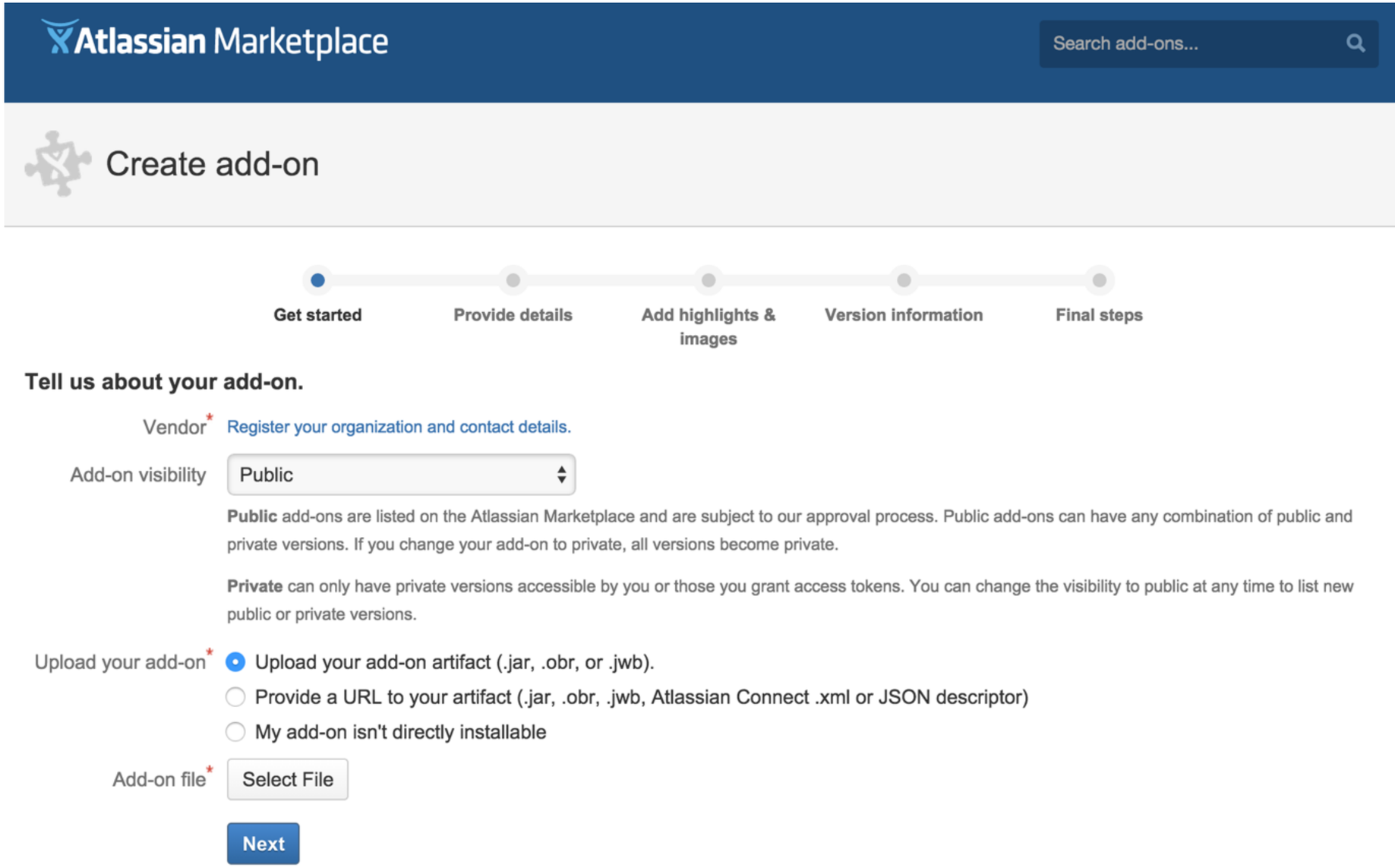
- HTTP callback to a URL
- Filtered by JQL
- When specific event occurs

```
"modules": {
  "webhooks": [{
    "event": "jira:issue_created",
    "url": "/issue-created"
  }]
}
```



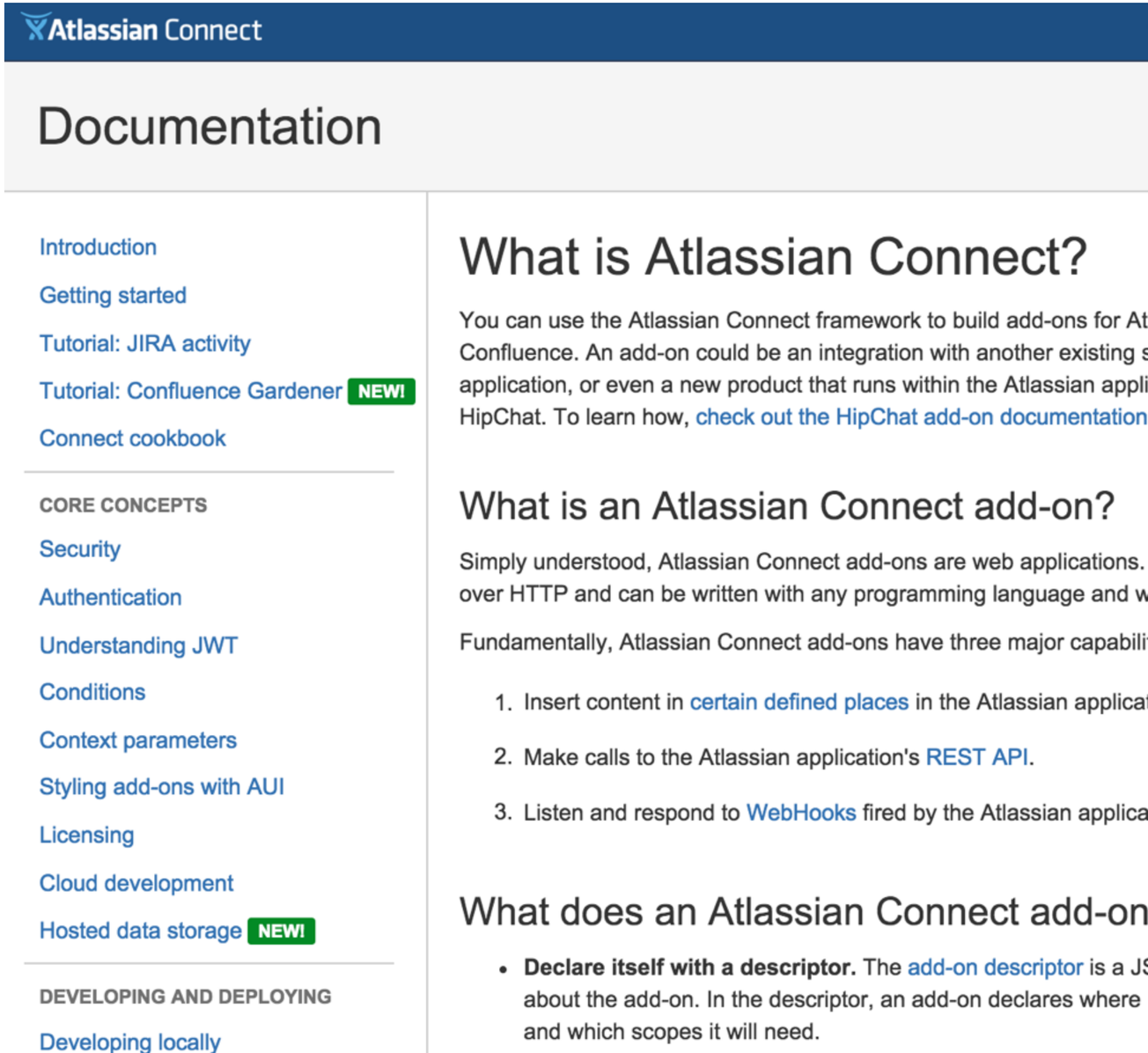
Register on the Atlassian Marketplace

- **Find**: in context catalog of add-ons
- **Try**: manage leads
- **Buy**: manage sales and licensing



Developer Tools for Atlassian Connect

- Vagrant Box for local development
- JSON descriptor validator
- JWT decoder
- Marketplace wizard to create a Connect listing
- Access Token model for private plugins
- Webhook Inspector



Pros

- Code in any language
- Use any build & dev tools
- Use anything: databases, search indexes, caching services, etc

Cons

- Host and scale a service
- No automated devloop (P2 SDK)
- Provide everything: databases, search indexes, caching services, etc

Why it’s awesome!

BYOS (Bring your own stack)

More services, than products

Sandboxing

UI Design

Aligned with microservice thinking

Why it sucks..

More interfaces must be stable

More developer tools, not just docs

Harder to create concrete examples

Complex to troubleshoot

Remember SPIs?

AC

P2

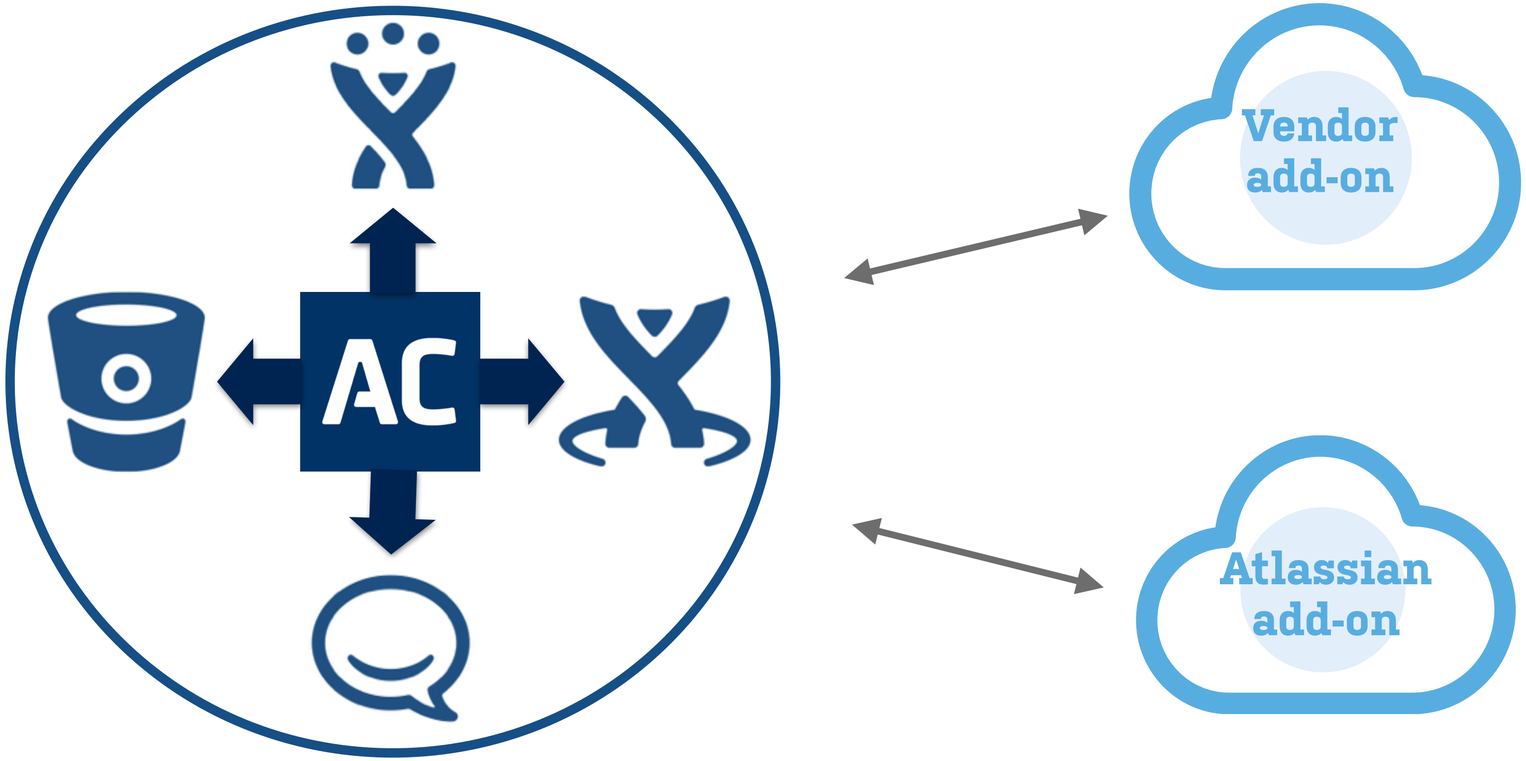
P1

open(ish) source

What's Next?

Consolidate and expand

- Extend Bitbucket and HipChat
- Build integrations between Atlassian applications
- Build complex vertical apps: powered by Atlassian



Questions?

A couple of Atlassians who are passionate about plugins