When I grow up, I want to be a platform
- Jira
- Confluence
- Bitbucket
- Trello
- Opsgenie
- Statuspage
Atlassian started Atlassian Cloud v1 (Jira Studio) in 2008. Bitbucket, HipChat were added in 2010-12. Stride, Statuspage, Trello, Opsgenie were added in 2017-18, and today, Atlassian offers a wide range of cloud platform services.
Agenda

Why

When

How
WHY
It will be better for everyone

You get to focus on your business logic!

Take one for the team!

You will get efficiency gains

A team of specialists will own this

You will unlock these business cases

This will allow for seamless UX

We built this, so you should totally use it
It will be better for everyone
Take one for the team!
We built this, so you should totally use it

You will get efficiency gains
You get to focus on your business logic!
You will get efficiency gains
A team of specialists will own this

You will unlock these business cases

This will allow for seamless UX
Users will get more consistent behavior
This will allow for seamless UX.
Users will get more consistent behavior.

You will unlock these business cases.
You get efficiency gains.

A team of specialists will own this.

Better for others, Biz value, Efficiency gains, Consistency.

It will be better for everyone.
Take one for the team!

We built this, so you should totally use it.
You get to focus on your business logic!

You will unlock these business cases.
Efficiency gains
A team of specialists will own this

Consistency

Better for others
Biz value
Efficiency gains
Consistency
Consistency
Biz value
Efficiency gains
Better for others

1st
Product Platform

2nd
Consistency
Platform Services

3rd
DFL
Foundations

Better for others
“RIGHT” ISN’T ENOUGH. WHEN BUILDING A PLATFORM, YOU NEED RIGHT + FEASIBLE + DESIRABLE.
WHEN
IT’S ALL ABOUT YOUR CUSTOMERS

- Shelving
- Early adopter
- Deferred Platform
DEFERRED PLATFORM

- ✔ clear platform signal
- ❌ catchup game
- ❌ no value for legacy consumers
EARLY ADOPTER

✅ clear requirements
✅ low producer waste

⚠ overfitting
⚠ consumer wait time
⚠ little usage
SHELVING

✅ no waste/wait for adopters

✅ blank slate

⚠ requirement mismatch

⚠ little/no usage
BACK TO THE TIMELINE...

proactive

Early adopter

Deferred Platform

reactive

Shelving

sweet spot!
HOW
How: breakdown

Plan

Build

Drive adoption
Telltale signs
Signals to be aware of and how to get them ASAP

Engagement patterns & anti-patterns
Over-the-wall, inner sourcing and others

Plan
Engagement patterns & anti-patterns

Over-the-wall, inner sourcing and others

Telltale signs
Signals to be aware of and how to get them ASAP

Plan
**EARLY SIGNS**

- Business strategy alignment
- Capability mapping
- Architecture forums FTW
- Innovation projects

**BAD SMELLS**

- 5 stages of grief in platform adoption
- Don’t want to build it, so it’s a platform thingie
Engagement patterns & anti-patterns

Over-the-wall, inner sourcing and others

Plan

Telltale signs
Signals to be aware of and how to get them ASAP
Engagement anti-patterns

Product turned platform
Priority clashes ruining best intentions

Over-the-wall
I build it, you run it.
Engagement patterns

Catchup
Addressing a tough platform sell.

Innersourcing
Planned, transitional ownership.

Venture bet
Address the unknown without any BS.

reactive

proactive
CATCH UP CASE STUDY: RATE LIMITING SERVICE

Legacy: local rate limiting
CATCH UP CASE STUDY: RATE LIMITING SERVICE

Modern: centralized rate limiting

- Client
- API gateway
- RL sidecar
- RL service
- API
INNER SOURCING CASE STUDY: PERMISSIONS SERVICE

Phase 1: Build in product

Consumer + Platform Partnership to define APIs and data model
INNER SOURCING CASE STUDY: PERMISSIONS SERVICE

Phase 2: Extract into platform

Overfitted data model caused problems
INNER SOURCING CASE STUDY: PERMISSIONS SERVICE

Phase 3: Re-write
Element or shared service
Immediate benefit vs longer term tight integration

Tailoring for better fit
Keeping platform reusable and extensible

Build
Element or shared service
Immediate benefit vs longer term tight integration

Tailoring for better fit
Keeping platform reusable and extensible

Build
**ELEMENT**

- e.g. UI libraries, mail service
- Drop-in and go
- Consistency & Engineering efficiency
- No cross-consumer integration

**SHARED SERVICE**

- e.g. Login, Marketplace
- More extensive integration
- Focus on biz value
- More consumers, more benefits
CASE STUDY: CROSS-PRODUCT EDITOR

Learn to surf in Hawaii

We suppose that a transferring power from the bio-regenerative subspace phaser array to the sub-evasive drive could rectify the problem.

```xml
<Router>
  <Switch>
    <SecureRoute entry="000342xR"/>
    <Route BEGIN
      <path id="circuitAB" d="M 100 3 50 1 50 -300"/>
      <path id="circuitBC" d="M 250 50 1 150 300"/>
    </path>
  </Route>
  <path d="M 175 200 1 150 0"/>
</Switch>
</Router>

cc Gil Admin
```
CASE STUDY: CROSS-PRODUCT EDITOR

Phase 1: Element
- Product UI
  - Editor UI
  - Product storage

Phase 2: Shared service
- Product UI
  - Editor UI
- Product 2
  - Editor UI
- Content storage
  - Migrate?
  - Notifications
  - Cross-product links
Element or shared service
Immediate benefit vs longer term tight integration

Tailoring for better fit
Keeping platform reusable and extensible

Build
Tailoring for better consumer fit

No tailoring
Just one way means probably no one can use it.

Fully customizable
Maintenance and extension nightmare with per-consumer edge cases
Tailoring for better consumer fit

No tailoring
Just one way means probably no one can use it.

Extensibility
Envision future consumers
Start simple with configuration
Design for pluggability (callbacks, events)

Fully customizable
Maintenance and extension nightmare with per-consumer edge cases
CASE STUDY: ECOSYSTEM PLATFORM CORE CONCEPTS

Diagram:
- Host defines Extension Point
- App is installed at Installation Context using

Legend:
- Host
- App
- Extension Point
- Installation Context
CASE STUDY: ECOSYSTEM PLATFORM EVOLUTION

- Libraries in serverland: High coupling, Low latency
- Services in cloud: Low coupling, High latency
- Serverless in cloud: Medium coupling, Low latency

Extension points | Lifecycle mgmt | Installation contexts | Admin | APIs
Carrots and sticks
Uniform adoption and team autonomy can work

Platform Essentials
Avoiding the “too much platform” problem

Drive adoption

Scaling
Handling the stampede of consumers
Carrots and sticks
Uniform adoption and team autonomy can work

Platform Essentials
Avoiding the “too much platform” problem

Drive adoption

Scaling
Handling the stampede of consumers
Carrots and sticks
Uniform adoption and team autonomy can work

Platform Essentials
Avoiding the “too much platform” problem

Drive adoption
Scaling
Handling the stampede of consumers
CARROT

Obvious big value for consumers

Fragmented adoption

Adoption changes with priorities

e.g. new “startup” features

STICK

Top-down mandate

Goes against team autonomy

Mandate principles, not implementations

e.g. GDPR, compliance
Shared goals
Joint commitment to measurable outcome

Play as a Team
Good for consumers, great for the company and users

Build for future
Platform commits to enhancements that benefit everyone

Rinse & repeat
Repeat for all consumers for uniform adoption

Carrot stick

New product X users login with platform Identity in FY20

Users have a single credential across Atlassian

Internationalisation for all!

All products start migrating to platform Identity in FY20
Carrots and sticks
Uniform adoption and team autonomy can work

Platform Essentials
Avoiding the “too much platform” problem

Drive adoption

Scaling
Handling the stampede of consumers
TOO MUCH PLATFORM?
TOO MUCH PLATFORM?
Platform Essentials

Path through platform
Simple paths delivering incremental value, setting up dependencies for later

Single entry point
Consumers have one place to start, one group to negotiate shared goals.

Platform-wide view
Continually simplify dependencies and adoption across all platform components
STEP 1: DEFINING VALUE-BASED ESSENTIALS

- Signup & Login
- Move between products
- Consistent collaboration
- Single sign-on
- Unified account
- Sessions
- Permissions
- Editor
- Content services
STEP 2: MAP OUT INCREMENTAL DELIVERY

Signup & Login

Move between products

Consistent collaboration

Single sign-on

Unified account

Sessions

Permissions

Editor

Content services
STEP 3: SIMPLIFY DEPENDENCIES

Signup & Login
- Single sign-on
- Unified account

Move between products
- Sessions
- Permissions

Consistent collaboration
- Editor
- Content services
STEP 3: SIMPLIFY DEPENDENCIES

Signup & Login

Consistent collaboration

Single sign-on

Unified account

Move between products

Editor

Sessions

Permissions

Content services
Carrots and sticks
Uniform adoption and team autonomy can work

Platform Essentials
Avoiding the “too much platform” problem

Drive adoption

Scaling
Handling the stampede of consumers
Scaling platform services

Dev experience is #1
- Self-service docs, ref apps, “dev accounts”
- Reliability for consumer dev and production

Think like AWS etc.
- Cattle not pets
- Protect all consumers e.g. rate limits, capacity plans
- Consumer-accessible metrics

Start early!
- By the time you realize you need it, it’s too late
Business value #1
Faster and greater ROI and adoption

Partner with early adopters
Build the right platform at the right time.

Plan to scale
“Pits of success” to drive adoption, think like an AWS

Takeaways
Partner with early adopters
Build the right platform at the right time.

Business value #1
Faster and greater ROI and adoption

Takeaways

Plan to scale
“Pits of success” to drive adoption, think like an AWS
Partner with early adopters
Build the right platform at the right time.

Business value #1
Faster and greater ROI and adoption

Partner with early adopters
Build the right platform at the right time.

Takeaways

Plan to scale
“Pits of success” to drive adoption, think like an AWS
“RIGHT” ISN’T ENOUGH. WHEN BUILDING A PLATFORM, YOU NEED RIGHT + FEASIBLE + DESIRABLE.
Thank you!

DIOGO LUCAS

SIDNEY SHEK
Rate today’s session

Cyberconflict: A new era of war, sabotage, and fear

We’re living in a new era of constant sabotage, misinformation, and fear, in which everyone is a target, and you’re often the collateral damage in a growing conflict among states. From crippling infrastructure to sowing discord and doubt, cyber is now the weapon of choice for democracies, dictators, and terrorists.

David Sanger explains how the rise of cyberweapons has transformed geopolitics like nothing since the invention of the atomic bomb. Moving from the White House Situation Room to the dens of Chinese, Russian, North Korean, and Iranian hackers to the boardrooms of Silicon Valley, David reveals a world coming face-to-face with the perils of technological revolution—a conflict that the United States helped start when it began using cyberweapons against Iranian nuclear plants and North Korean missile launches. But now we find ourselves in a conflict we’re uncertain how to control, as our adversaries exploit vulnerabilities in our hyperconnected nation and we struggle to figure out how to deter these complex, short-of-war attacks.

David Sanger
The New York Times

David Sanger is the national security correspondent for the New York Times as well as a national security and political contributor for CNN and a frequent guest on CBS This Morning, Face the Nation, and many PBS shows.