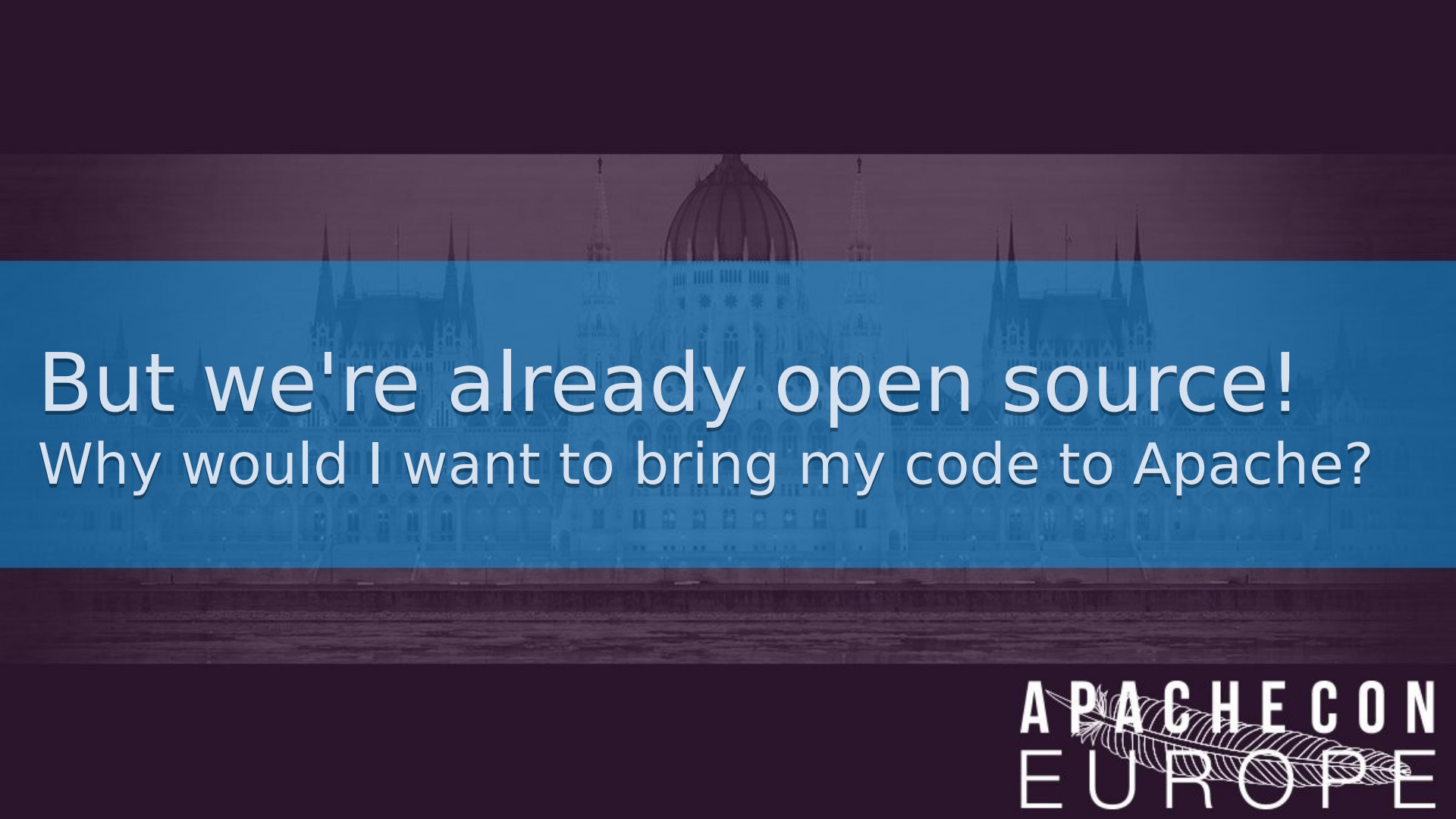

But we're already open source!
Why would I want to bring my code to Apache?
APACHECON
EUROPE

Nick Burch

CTO, Quanticate

Open, Open and Open!

- Open vs Open Source
- The ASF – What's the deal?

- Case Study 1 – Alfresco and Apache Tika
- Case Study 2 – Alfresco and Apache Chemistry
- Case Study 3 – Alfresco and Activiti

- The Benefits
- And the downsides...

Open vs Open

APACHECON EUROPE

Open Source

“denoting software for which the original source code is made freely available and may be redistributed and modified”

- You can get the source behind a program (if you want)
- You can study it, learn from it, understand it, debug it
- You can make changes to it (or pay someone else to)
- You can use your new version without having to pay
- You can share your changes and improvements
- There is a license, but it's a distribution not use one

Open Development

- Open Source is all about the software today
- Open Development is about the software tomorrow
- Quite possible to be Open Source but Closed Development

- Who decides what the priorities are?
- Where do those discussions happen, and are they shared?
- Who can influence the direction of the project?
- Who can contribute (directly or indirectly)?
- Who dictates releases, “done” etc?

Copyright Assignments

- Applies when you want to contribute a fix or enhancement
- Assignement requires you sign over your rights to your changes to another entity, to do with as they wish
- Most common with Commercial+CopyLeft

- Can be used “for good”, to permit relicensing in future, and to allow a central entity to fight infringement claims
- Can be used “for money”, to allow a company to license out your changes to their enterprise customers
- Typically very divisive, problems if not accepted

CLAs (License Agreements)

- At first glance, these look quite similar
- Legal agreement about a contribution
- But... Very different kettle of fish!

- Copyright Assignment requires you hand over your rights
- CLAs are about ensuring you understand the open source license, and are allowed to contribute under it
- “License says you can have this, and I confirm you can”
- Paper trail of contributions
- Defends against “but that's *our* code!”

Community Direction

- Within any project, whether open or closed, proprietary or shared, there's a process to decide on direction
- When will the next release be considered “done”?
- What features should be worked on? Not worked on?
- What's the vision for the project? What's out of scope?
- Who can join in? Who can contribute?
- How are new changes reviewed+accepted? And by who?
- How are conflicts and debates handled? Who has a say?
- How can this be changed over time?

Licensing

- All Open Source licenses are distribution licenses
- Use of the software is free (support + add-ons may not be!)
- To broad families of Open Source licenses
  - Copyleft (eg GNU-GPL, LGPL)
  - Permissive (eg Apache License, BSD, MIT)
- Strong philosophical differences divide, plenty of license flame-wars online debating all of these!
- Have different restrictions on building on top of them
- Influences the business models you can build on top

Many Models

- There isn't a “One True Way”
- But the ASF has a “often least worst way!”

- More on the ASF shortly
- More on pluses and minuses later

- No one model is “always right” or “always wrong”, need to weigh up the benefits and downsides
- There is help for weighing this all up!

The Apache Software Foundation

APACHECON EUROPE

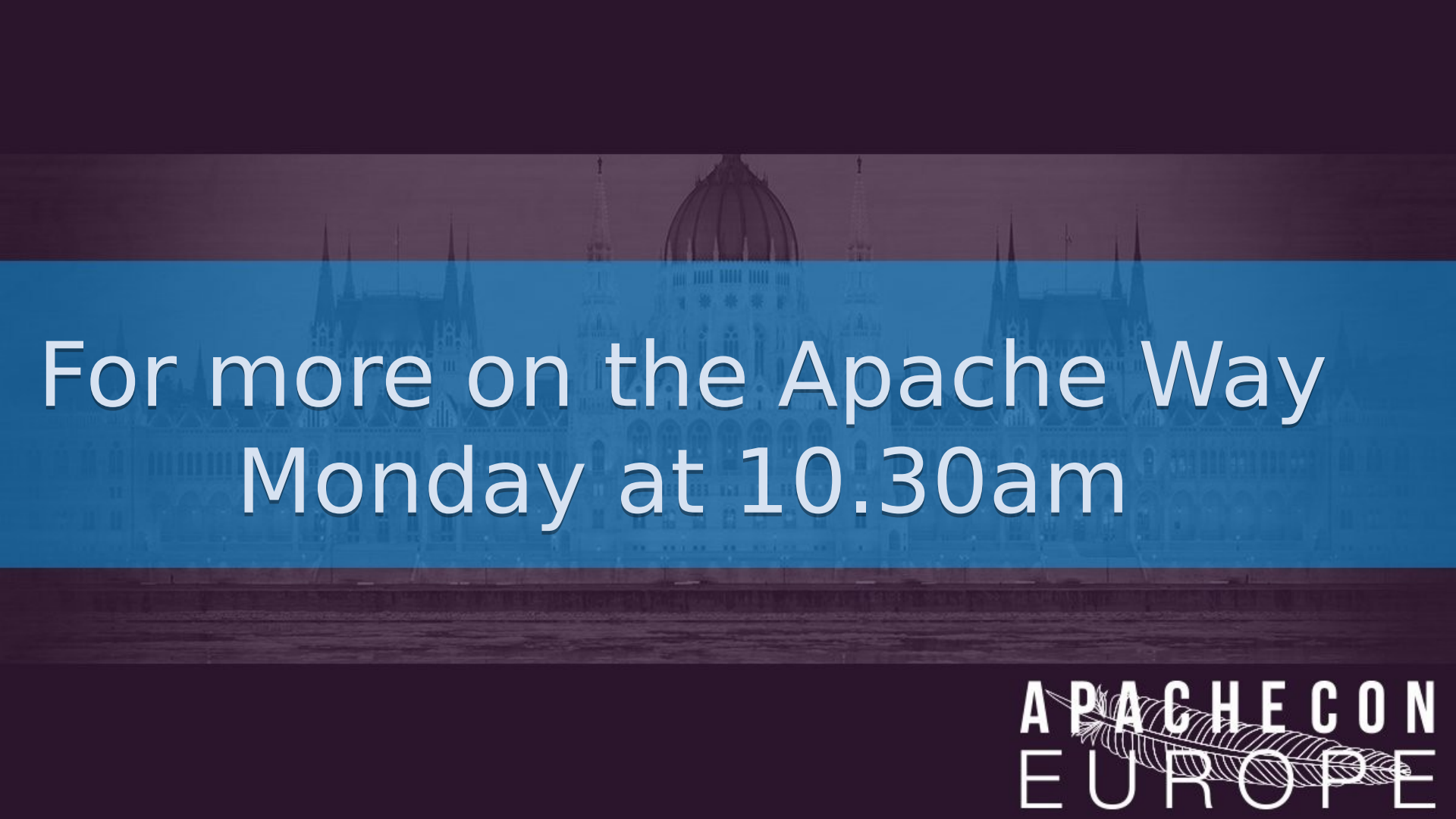
What is the ASF?

- US 501(c)3 not-for-profit foundation
- Founded in 1999 with one project (webserver)
- Today has over 150!

- Meritocratic, community driven open source
- Anyone can get involved, and not just as a coder
- Decisions taken by the community
- Work done by volunteers
- Open Development, friendly to business involvement

Driven by projects

- A number of projects
- Each project is responsible for their own code, community and direction
- Board provides oversight, but that's it
- Board has no say on what code gets written, nor what direction projects take, nor what projects we should start. All of that is devolved to the projects themselves
- Foundation has some common support (eg infra, press, trademarks), to help projects focus on their code and on their communities


For more on the Apache Way
Monday at 10.30am
APACHECON
EUROPE

Case Study 1: Alfresco and Apache Tika

Alfresco

- Open Source Enterprise Content Management
- >1 million active users, >4 million downloads, plus Cloud
- Open Source, but not Open Development (mostly)
- Written mostly in Java and JavaScript
- ECM – Capture, Manage, Store, Preserve, Deliver, Search, Collaborate, within and between organisations
- Includes Document Management, Records Management, Asset Management, WCM, Workflow etc
- Alfresco is Open Source, and builds on lots of of Open Source projects, especially from Apache and Spring

Metadata & Transformers

- Content Transformers and Metadata Extractors are core Alfresco services, present from the very start
- Transformers are used for full text indexing, web previews, icons, web friendly versions etc
- Metadata Extraction used to synchronise the document properties with the repository
- Both normally used with uploaded binary files
- eg upload a Word Document, text extracted for indexing, PDF version generated, title + author extracted

Alfresco 3.3 - Formats

- PDF
- Word, PowerPoint, Excel
- HTML
- Open Document Formats (OpenOffice)
- RFC822 Email
- Outlook .msg Email
-
- And that's it!

Apache Tika tika.apache.org

- Apache Project, started in 2006
- Initially grew out of Lucene, but now very widely used
- Provides detection capabilities – eg this is a Word Doc
- Parsers for a very wide range of formats
- Extracts metadata, and provides a consistent model, eg created by vs author
- Textual content available as Plain Text and XHTML
- Hides complexity of file formats and librarys, presents a simple and powerful API across all of them
- Easy to use and extend

Sharing process

- Identify upstream projects (mostly Apache Tika + POI)
- Get the latest versions of the code
- Compared Alfresco code to upstream, and identify areas where Alfresco did more / upstream didn't handle
- Wrote unit tests for things Alfresco did better, to avoid regressions after the push upstream
- Identified which bits of upstream would be improved, and what bits would be whole new features
- Worked with upstream projects, shared ideas for enhancements, checked hadn't missed things

Sharing process continued

- Produced patches, submitted to the Apache projects
- Get feedback and improvements on patches
- Gained trust of communities, granted commit rights
-
- Once most things upstream, update Tika and POI versions used in Alfresco, check everything worked
- Wrote Tika wrappers for Alfresco
- Converted Alfresco code to call Tika
- Used unit tests to ensure no lost functionality
- Enabled other formats via Tika

Alfresco 4.2 - Formats

- Audio – WAV, RIFF, MIDI, MP3, MP4, Ogg Vorbis, FLAC
- CAD – DWG, PRT
- EPub
- Feeds – RSS, Atom
- HTML, XHTML, XML
- Images – JPEG, GIF, PNG, TIFF, Bitmap
  - Incuding EXIF data where present
- RFC-822 MBox email
- Microsoft Outlook .msg email

Alfresco 4.2 - Formats

- IWorks (Keynote, Pages, Numbers)
- Microsoft Office (Binary) – Word, PowerPoint, Excel, Visio, Publisher, Works
- Microsoft Office (OOXML) – Word, PowerPoint, Excel
- Open Document Format (OpenOffice)
- PDF
- RTF
- Plain Text
- Scientific Data – CDF, HDF

Alfresco 4.2 - Formats

- Archive - Zip, Tar, Tar-GZ, Tar-BZ2, Compress, Ar
- FLV Video
- MP4 Video
- Java Class Files
- CHM (Windows Help)
- Executable Libraries and Programs
- Configurable external programs
-

And probably some others I've forgotten

Alfresco 5.0 – New Formats

Thus far, Alfresco 5.0 is set to get the following new formats, on release, at no extra effort beyond upgrading Apache Tika:

- Archive - 7z
- Audio – Ogg Opus and Speex
- Video – Ogg Theora
- Image – Tesseract OCR support
- Source Code
- Email – Outlook PST
- Font – Adobe Font Metrics

And maybe more before 5.0 Enterpise Final is released!

Case Study 2:

Alfresco and Apache Chemistry

APACHECON EUROPE

CMIS (OASIS Standard)

- CMIS – Content Management Interoperability Services
- Standard for talking to content repositories
- Provides a standard way to login, navigate, search, upload, version, delete, change permissions, alter properties, set relationships, query etc
- Supported by a wide range of content repositories
- Adobe, IBM, Microsoft, Alfresco, SAP, EMC, HP, Liferay, Nuxeo, OpenText, Oracle, and some others
- CMIS 1.0 released in 2010
- CMIS 1.1 released in late 2012

First Alfresco Support

- Alfresco was involved in CMIS from an early stage
- First Alfresco CMIS support was added before the CMIS 1.0 spec was completed, and was used to feed back on the proposed standard
- First support was developed internally, took quite a while, and turned out to have some issues
- Desire for a pure-Java server implementation, which would be more rhobust and easier to test + validate
- Desire for a CMIS client for a few different platforms
- Desire not to have to do it all in-house

Two Rival Camps

- Alfresco, OpenText and SAP started working on an open source Java client and server as OpenCMIS
- Nuxeo, Adobe and friends started on Apache Chemistry
- Both had strengths and weaknesses
- Quite different development and governance approaches
- Some lack of trust between camps
- Quite a lot of misunderstandings about what Apache and the Apache Way was, and worry about a loss of control or direction, from some in OpenCMIS
- Then an idea to contribute OpenCMIS as another project

The Joining Together

- Lots of debate on mailing lists and in private
- Eventually a get-together in Munich, coupled with lots of feedback to the mailing list, and rattification there
- Outcome was a merged project, Apache Chemistry, which combined the best of both
- Lots of activity, lead quite quickly to a CMIS-1.0 compatible test server, server framework, client and test suite
- Took some work to implement this new server into Alfresco, but it proved better than the in-house one
- Initially only a handful of companies used it, rest tested

Going Forward

- Alfresco received lots of bug fixes “for free” just from upgrading the OpenCMIS libraries
- More companies started basing their CMIS implementations on Apache Chemistry, having realised that it was both high quality and already available
- Virtuous circle of enhancements and fixes
-
- CMIS 1.1 support added to Apache Chemistry mostly by SAP, with help from others
- Adding CMIS 1.1 support to Alfresco took a tiny fraction of the time CMIS 1.0 did, due to Chemistry support

Case Study 3: Alfresco and Activiti

APACHECON EUROPE

BPMN-2.0 Workflows

- BPMN – Business Process Modeling Notation
- Way of describing business processes (workflows), which can be displayed visually, but also executed
- Covers most of the key elements of a workflow
- Ideally can be created by business analysts
- Big step on from previous BPMN versions is that the format is executable – the same thing that the analyst creates / changes is what gets run
- Industry standard for doing this (previously each engine had its own way of doing that)

Alfresco and Workflow

- Alfresco Workflows were based on jBPM
- In 2010, jBMP was LGPL and owned by Redhat
- Alfresco were finding some problems with jBPM that were hard to fix, and seemed to be architectural
- jBPM core devs were interested in a re-write, Redhat less so
- Alfresco was trying to reduce the number of copyleft dependencies in the codebase, jBMP was a big part
- Alfresco had moved from Hibernate to iBatis, and jBMP remainging with Hibernate was making development + support harder

The Birth of Activiti

- Alfresco hired two of the core jBPM developers, Tom Baeyens and Joram Barrez, plus some others
- Charged with developing an Apache licensed, open source implementation of BPMN 2.0
- Alfresco was the be the “launch customer”, but being open source anyone can (and hopefully would) use it too
- Needed some changes to Alfresco to make the the Workflow APIs (already pluggable) work well with Activiti as well as jBMP
- Aim was for Activiti to be the default engine, and more powerful / richer than jBPM was

ASL v2, but not ASF

- License was picked as Apache License v2
- However, after lots of internal debate, decision was made not to submit the project to the Apache Incubator
- Alfresco wanted their next release to ship with Activiti, meant they had strong pressures to hit a ship date
- Initial feature set largely dominated by what was needed for Alfresco, other features were added but mostly after the first release
- Tom Baeyens is a “BPM-God”, really knows his stuff, but after Redhat frustrations wanted to drive it
- Initially not good Incubator fit, later why change?

Benefits

Shared Maintenance

- By sharing upstream, your support overhead is shared
- Many companies, organisations and individuals can join in and contribute towards the project
- When others extend your project, you benefit for free
- More users means greater chance of community enhancements / new features
- More users means more bug reports, but also more bug reports that come with fixes (patches)!
- Lower per-organisation maintenance costs
- Often faster development

Better Bug Reports

- People familiar with Open Source often know how to report bugs better
- By sharing your code, more people can use it, so there's a wider pool of people to test and report on it
- “I have found a bug, but I can't share the details of it” is less common, and more chances of a duplicate which can
- More likely that another user can come along and fill in the details of existing bugs
- If you're open and other companies are joining in, greater chance someone will have dedicated testers!

New Ideas, New Features

- When other companies feel they can base their business on your project, then they can dedicate more resources
- When individuals feel they have a say, they're more likely to spend their time working on it
- More developers means more ideas for fixing problems, enhancing exisiting code, and adding new things
- When contributions are felt to be welomed and appreciated, more likely for people to share back
- Can be small changes over time, or can be large new features, all for (almost) free!

Learning from the best

- The world is a very big place
- No matter how great you are, there's someone out there who's smarter than you are!
- And many more people who've been there before
-
- If you have more contributors, there's more chance of someone improving your idea or code
- By seeing how they do it better, you not only get a better project, but learn from it to improve yourself
- Collaborate to get the best of everyone

Diverse Users and Devs

- We are often a bit of a mono-culture
- Some worse than others.....
-
- If you draw from a very small pool, you'll have people who tend to think the same, and will be blinded by that
- If you have people from all around the world, from all walks of life, you've more chance of diverse opinions
- Best time to fix a mistake is at the start
- Even if it's just “hmm, that breaks when you're not +00:00” !

Community and Goodwill

- Sharing and being open can bring positive press
- Your organisation can gain recognition from their involvement
- Halo effect – share in the buzz from the project
- And an open project with lots of people has more buzz!
-
- If you're a known good team player, it's easier to call in favours from the community (eg critical bug fix)
- If you're known generally as a good team player, easier to get help or support in related projects (eg dependencies)

Downsides

Lack of Control

- If it's your project, you can do with it as you want
- Very easy to make decisions, get things done
- If it's an ASF project, you need to explain your ideas, and convince the rest of the community of why
- If it's in-house, you can have a quick chat to decide
- If it's an ASF project, you need to write it all down, explain it, and wait to allow everyone to have a chance to review
- Management understand traditional projects
- Can lead to confusion and hard questions if not properly explained, expect some “what do you mean?”s

Release dates

- If it's your project, you have full control (and responsibility) about “done”, and the associated release dates
- If it's an ASF project, it's a community decision
- You can't “demand” that someone else fixes a bug
- If the community is all excited and working on a new feature, it may not be the right time for a stable release
- If you're on a different cycle to many other contributors, expect to still have to do much of the “boring bits”
- Need to get the community to share release vision

Direction / Features

- If it's your project, you decide what's in, you direct the developers, off they go (or fail trying....)
- If it's an ASF project, you need to explain the idea to the community, and get their buy-in
- You may still then have to do most of the work on it – you can't force the volunteers to work on something!
- Your vision for where it should go may not match others
- Your “value add” might end up getting re-done by the community, and you can't stop that
- Can confuse management, who expect a different dynamic about getting new things in

Marketing & Branding

- If it's your project, you can brand it how you want, associate your company with it how you want
- ASF has trademark rules which apply, you mustn't confuse your product with the project
- Need to co-ordinate with the rest of the community on branding, press releases, conferences etc
- Can be hard to get agreement on these things, can be prone to bike-shedding if not careful
- Potentially big loss of control

In Summary

APACHECON EUROPE

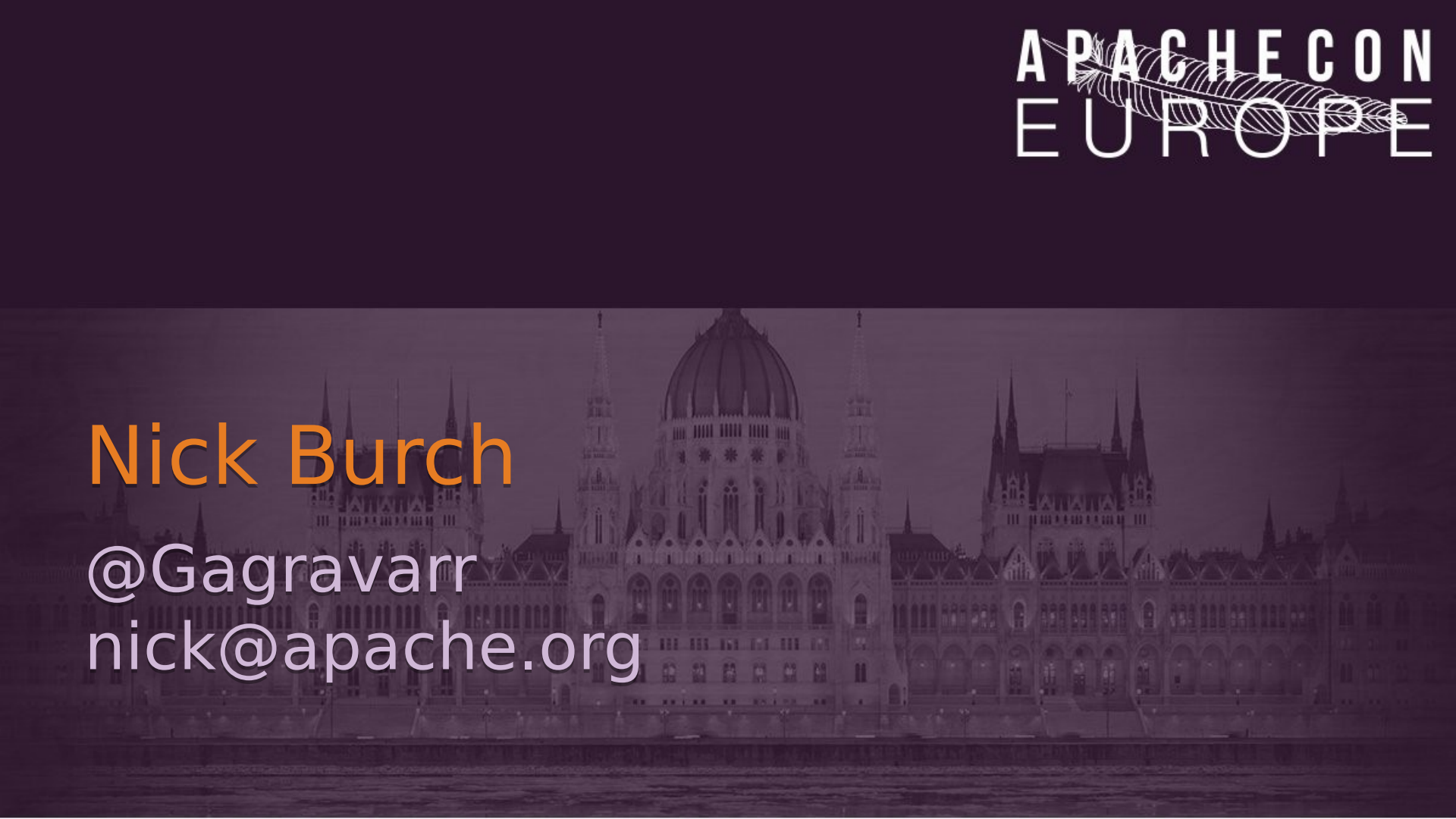
Why it's Good

- More involvement
- Better bug reports, wider testing
- Other companies (including competitors) much more willing to join in and share resources
- Learn from others, and collaborate with them to improve
- More diverse community can mean problems get spotted and fixed sooner (+cheaper)
- More opportunities from wider use
- New features and fixes “for free”

But not for everyone!

- You're no longer in charge!
- Need to be willing to give up some control in order to welcome in the wider community
- Not instant – will be an up-front cost, you need to be in for the longer term to take advantage
- Need to train up your marketing team, and management
- Won't all be smooth sailing, there will be problems
- BDFLs aren't allowed
- “No jerks”, you need to work with the community


Any Questions?
APACHECON
EUROPE



Nick Burch

@Gagravarr

nick@apache.org