LinuC レベル1 Version10.0
技術解説無料セミナー
2020/08/01 開催

オープンソースの文化を知ろう

本日の講師

ワイズプランニング
吉田　行男

LPI-JAPAN

自己紹介

■ 【経歴】
- 入社当時は、金融端末のソフトウェア開発に従事。
- 2000年頃から、Linux/OSSのビジネス開発を担当。
- 2012年から、オープンソース専門組織に所属。
- 2019年、定年退職により独立。

■ 【現在の業務】
- OSSを活用したビジネス構築のための支援
  - 新しい技術/OSSの発掘・評価検証
  - ビジネス・ソリューションの立ち上げ支援
- OSSコンプライアンス管理
  - ガイドライン作成、社内プロセス構築支援
- 各種講演、執筆

#LinuC学習中

(*)2018/11：北東アジアOSS推進フォーラムにて「OSS貢献賞受賞」

LinuCレベル１／レベル２ Version10.0とは

#LinuC学習中

■LinuCとは

クラウド時代の即戦力エンジニアであることを証明するLinux技術者認定資格

✓現場で「今」求められている新しい技術要素に対応

- オンプレミス／仮想化を問わず様々な環境下でのサーバー構築
- 他社とのコラボレーションの前提となるオープンソースへの理解
- システムの多様化に対応できるアーキテクチャへの知見

✓全面的に見直した、今、身につけておくべき技術範囲を網羅

今となっては使わない技術やコマンドの削除、アップデート、新領域の取り込み

✓Linuxの範疇だけにとどまらない領域までカバー

セキュリティや監視など、ITエンジニアであれば必須の領域もカバー

■Version10.0と従来の出題範囲の比較

#LinuC学習中

	テーマ	Version 10.0	従来
LinuC-1	仮想技術	・仮想マシン／コンテナの概念 ・クラウドセキュリティの基礎	← （Version10.0で新設）
	オープンソースの文化	・オープンソースの定義や特徴 ・コミュニティやエコシステムへの貢献	← （Version10.0で新設）
	その他	→ （Version10.0で削除）	アクセシビリティ、ディスククォータ、プリンタの管理、SQLデータ管理、他
LinuC-2	仮想化技術	・仮想マシンの実行と管理（KVM） ・コンテナの仕組みとDockerの導入	← （Version10.0で新設）
	システムアーキテクチャ	・クラウドサービス上のシステム構成 ・高可用システム、スケーラビリティ、他	← （Version10.0で新設）
	その他	・統合監視ツール（zabbix） ・自動化ツール（Ansible）	← （Version10.0で出題範囲に含む）
		→ （Version10.0で削除）	RAID、記憶装置へのアクセス方、FTPサーバーの保護、他

#LinuC学習中

今回のテーマ

オープンソースの文化を知ろう

OSSの歴史と定義

OSSの始まり

■OSSの始まり

#LinuC学習中

1969：AT&T ベル研究所にてUnix開発 教育機関向けにソースコードを無償公開

1983 :独占禁止法違反裁定によるAT&T分割に伴いUnixを製品化

1985： Free Software Foundation(FSF)設立
GNUプロジェクトが「GNU宣言」を発表

1991：Linux開発着手 ➡ 1994：Linux1.0 リリース

1997： 「伽藍とバザール」発表

1998：Mozillaのソースコードを公開

「オープンソースソフトウェア」誕生

ブラウザ戦争

1996 :MS Internet Explorerを無償化

1994 :インターネット商用利用開始

OSSの定義

■OSI(*1)が定めるOSSの定義

#LinuC学習中

オープンソースの権利

① 自由な再頒布が出来ること
② ソースコードを入手できること
③ 派生物が存在でき、派生物に同じライセンスを適用できること

オープンソースライセンスが備えるべき条件

④ 差分情報の配布を認める場合には、同一性の保持を要求してもかまわない
⑤ 個人やグループを差別しないこと
⑥ 適用領域に対する差別をしないこと
⑦ 再配布において追加ライセンスを必要としないこと
⑧ 特定製品に依存しないこと
⑨ 同じ媒体で配布される他のソフトウェアを制限しないこと
⑩ 技術的な中立を保っていること

ポイント

- オープンソース　≠　著作権を放棄されたソフトウェア
- ソースコードがインターネット等で公開されている
- 再配布の自由と改変の自由がある

※1・・Open Source Intiative(オープンソース文化の啓蒙を目的に設立された組織)

OSSのライセンス

コピーレフト(Copyleft)という考え方

■コピーレフトという考え方(FSFの行動原理)

- 著作物の利用、コピー、再配布、翻案を制限しない
- 改変したもの（二次的著作物）の再配布を制限しない
- 二次的著作物の利用、コピー、再配布、翻案を制限してはならない
- コピー、再配布の際には、その後の利用と翻案に制限が無いよう、
  全ての情報を含める必要がある（ソフトウェアではソースコード含む）
- 翻案が制限されない反面、原著作物の二次的著作物にも
  同一のコピーレフトのライセンスを適用し、これを明記しなければならない

#LinuC学習中

**Free(無償)ではなく、
Free(自由)**

ライセンスの類型

■ライセンスの類型

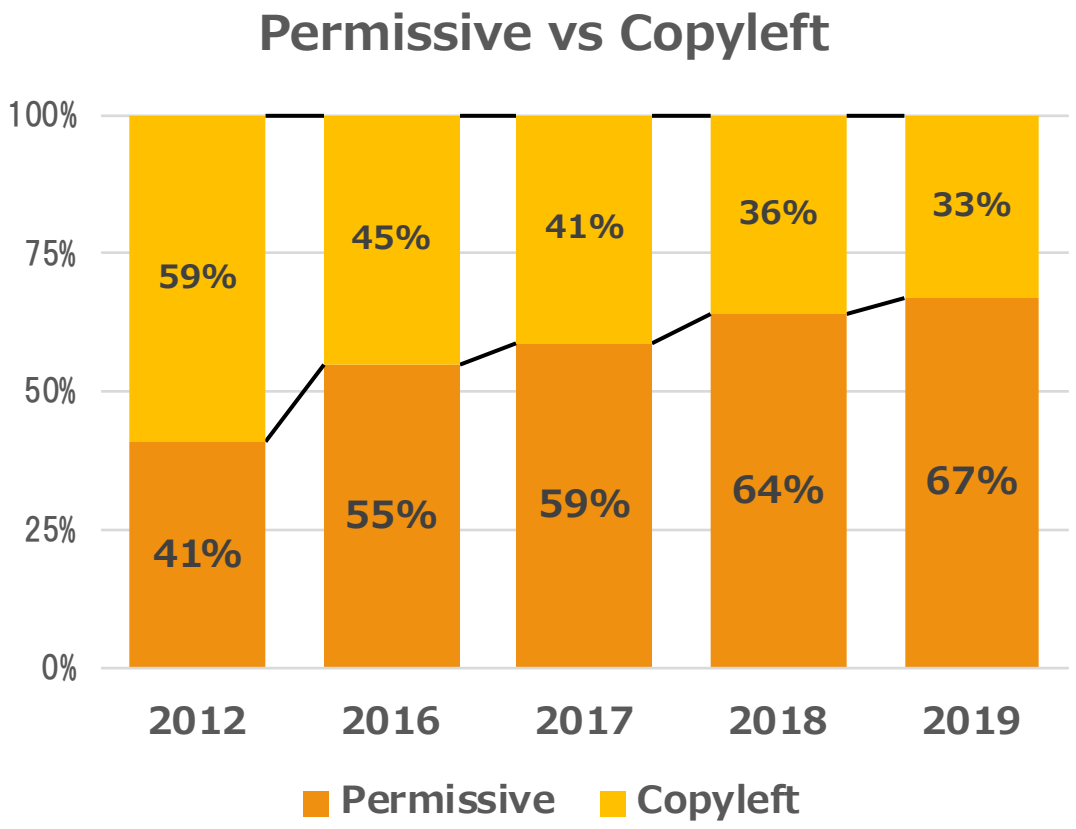
#LinuC学習中

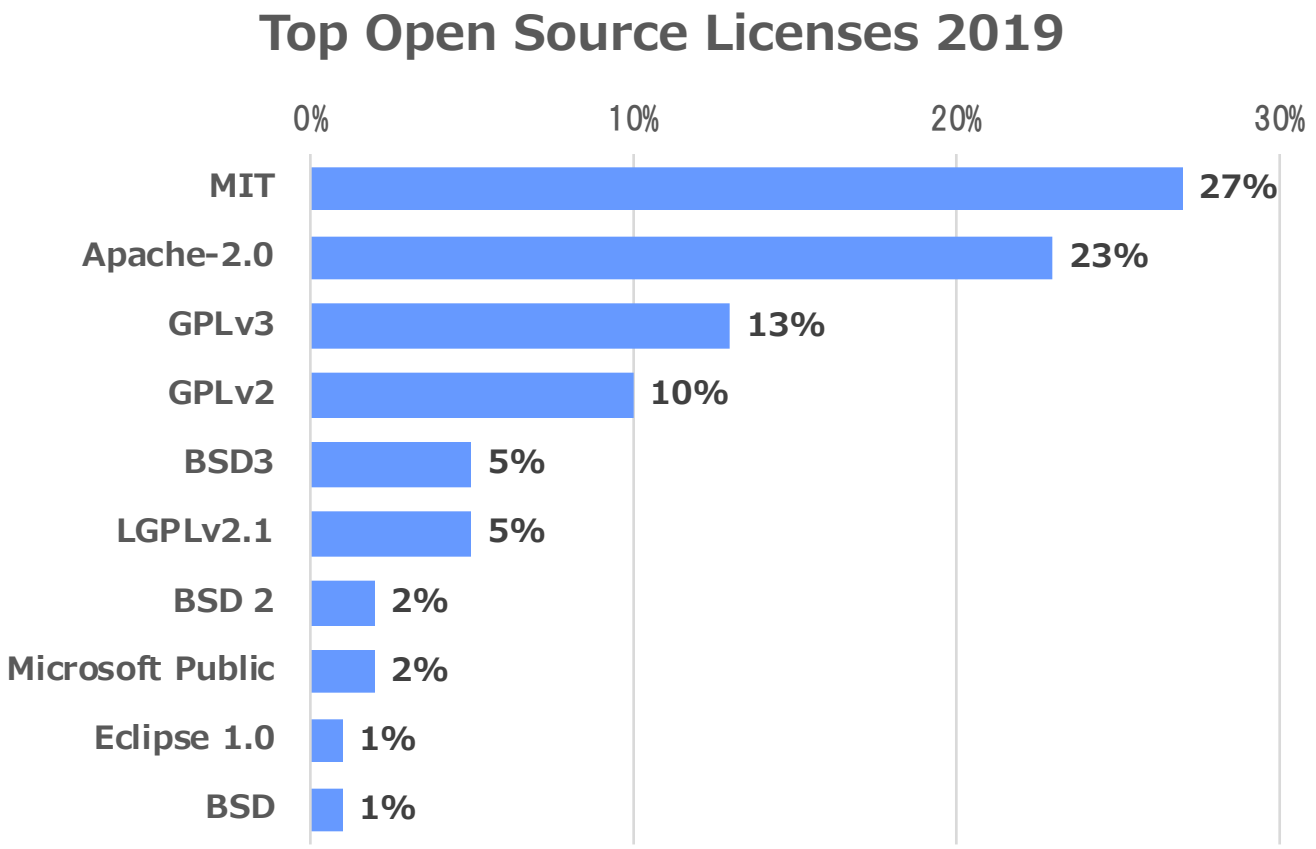
	類型		主なライセンス
1	コピーレフト型	➢ ライセンシの派生物にまで同じライセンスの適用を**要求する**。 ➢ ライセンサが配布するOSSをライセンシが他のソフトウェアと組み合わせた場合、 ライセンサはライセンシに組み合わせ先のソフトウェアにまで同じライセンスの適用を**要求する**。	GNU GPL GNU AGPL EUPL(European Union Public License)
2	準コピーレフト型	➢ ライセンサに派生物にまで同じライセンスの適用を**要求する**。 ➢ ライセンサが配布するOSSを、ライセンシが他のソフトウェアと組み合わせた場合、ライセンサはライセンシに組み合わせ先のソフトウェアまでは、同じライセンスの適用を**要求しない**。	GNU LGPL MPL (Mozilla Public License) Eclipse Public License CPL (Common Public License) IBM Public License Artistic License (Perl License)
3	非コピーレフト型 (Permissive型)	➢ ライセンシに派生物にまで同じライセンスの適用を**要求しない**。 ➢ ライセンサが配布するOSSを、ライセンシが他のソフトウェアと組み合わせた場合でも、ライセンサはライセンシに組み合わせ先のソフトウェアにまでは同じライセンスの適用を**要求しない**。	BSD License FreeBSD Copyright MIT License X11 License ZPL (Zope Public License) Apache License

ライセンスの適用状況①

■コピーレフト型が減少傾向、MITとApacheで約半数。

#LinuC学習中



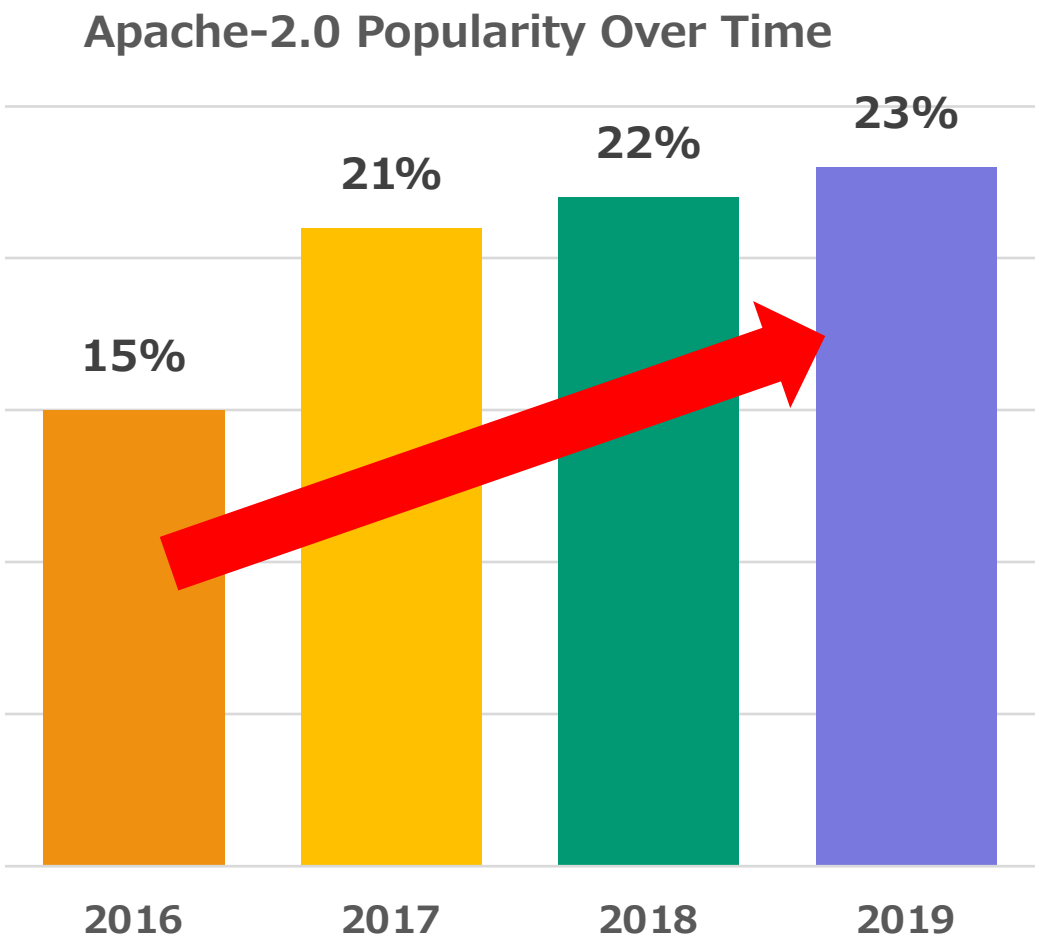


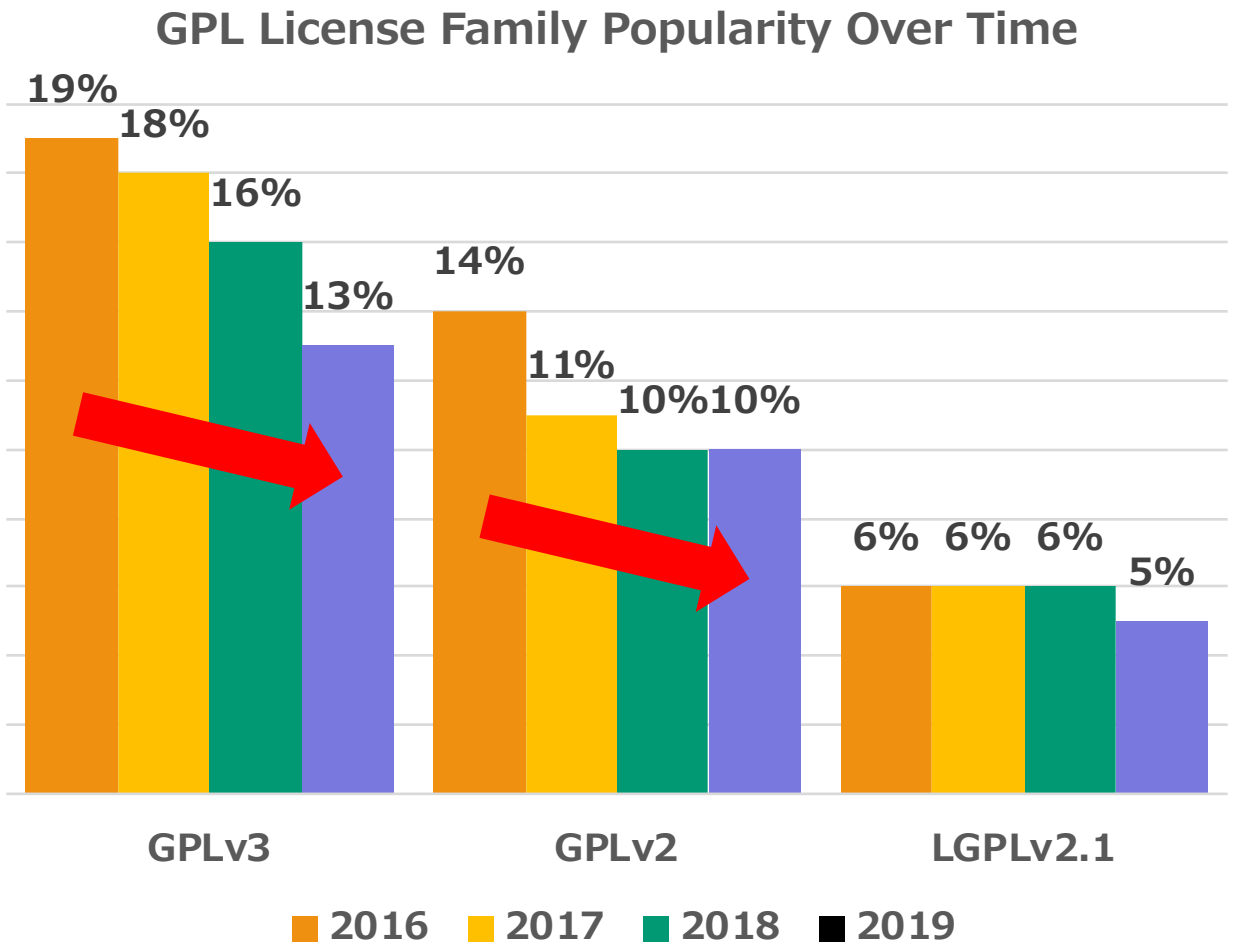
https://resources.whitesourcesoftware.com/blog-whitesource/top-open-source-licenses-trends-and-predictions

ライセンスの適用状況②

■Apache2.0が増加傾向、 GPL系の減少は止まらず。

#LinuC学習中





https://resources.whitesourcesoftware.com/blog-whitesource/top-open-source-licenses-trends-and-predictions

著名なライセンス紹介(①)

① GPLおよびLGPL（最新版は v3）

#LinuC学習中

- Free Software Foundation が作成したライセンス。
- GPLが適用された OSSを改変して派生著作物を作成し、配布する場合、その全体についてGPLの条件を遵守する義務あり。
  - GPLプログラムと他のプログラムをリンクや結合した場合、他のプログラムにもGPLの条件を適用する必要あり
- LGPL は、GPL の条件を少し弱めたライセンスで、LGPLが適用されたOSS（LGPL プログラム）と他のプログラムをリンクした場合、リンクした他のプログラムに LGPLを適用する必要はないが、リバースエンジニアリングを許諾する（禁止しない）義務あり。
- 最新のバージョンは GPLv3 、GPLv2 の OSS も多数存在。
  - GPLv3で追加された条件は、OSS で実施された特許権について、当該 OSS の開発者が保有する特許権を許諾する条件や、ユーザ製品に OSS を利用する場合には修正したOSS を再インストールできるように、「インストール用情報」を提供する義務などが追加された。
- また、LGPLv3 については、GPLv3 の追加的許諾条項となった。
- なお、GPL や LGPL は、OSS を配布した際に条件が課されるため、誰にも配布しない場合は、ソースコードの提供等の義務はない。これをカバーするためにAfferoGPLv3 が作成された。
- このライセンスは、クラウドサービス等のサーバでOSS（AfferoGPLv3）を利用した場合、（OSS のバイナリを配布しない場合でも）サーバにアクセスするクライアントへソースコードを提供する条件がGPLv3に追加されたライセンス。

著名なライセンス紹介(②)

#LinuC学習中

② CPL v1.0

- IBMが作成したライセンス。
- CPLv1.0が適用されたOSSを配布する場合は、ソースコードも提供する義務あり。
- CPLプログラムのコントリビュータに対する特許訴訟（訴訟対象は OSS に限定されていない）を制限。

③ EPL v1.0

- Eclipse Foundation が CPLv1.0 を基に作成したライセンス。
- 上記記載の特許訴訟の制限が削除。

④ MPL（最新版は v2.0）

- Mozilla Foundation で作成されたライセンス。もともとは Netscape Communications の弁護士により作成されたもの。
- MPLが適用された OSS（MPLプログラム）のソースコードを配布する場合は、ソースコードも提供する義務あり。
- 最新のバージョンは MPLv2.0。MPLv1.1 の OSS も多数存在。
- MPL v2.0 で追加された条件としては、GPL関連のOSSと組み合わせて配布する場合、“Secondary License”（GPLv2.0/LGPLv2.1/AfferoGPLv3.0、以降のバージョンを含む）を適用することを許諾するか否かを定めることが可能に。

著名なライセンス紹介(③)

⑤ Apache License（最新版は v2.0）

#LinuC学習中

- Apache Software Foundation が作成したライセンス。
- 最新のバージョンは Apache License v2.0 。Apache Software License v1.1も多数存在。
- Apache Software Licensev1.1は、ドキュメントへの謝辞の記載義務がある。
- Apache License v2.0 では、謝辞の記載義務は削除され、開発者による著作権や特許権の許諾が明確に。

⑥ BSD License

- カリフォルニア大学で作成されたライセンス。
- 制限は緩く、著作権表示と許諾リスト、免責条項を記載する義務あり。
- 以前の OLD BSD License（4-Clause）には、宣伝媒体に所定の謝辞を記載する条件があったが、NEW BSD License（3-Clause）では、削除された。2-Clauseの BSD License もあり、3-Clause から開発者名等の利用許諾に関する条件を無くし、ソースコードとバイナリの配布条件を記載したライセンス。

著名なライセンス紹介(④)

⑦ MIT License


#LinuC学習中

- マサチューセッツ工科大学で作成されたライセンス。
- 制限は緩く、著作権表示と許諾表示を記載する義務がある。
- BSD License（2-Clause）と利用条件は類似しているが、MIT License では、サブライセンスの許諾等、著作権者が許諾する内容が細かく記載されている。

⑧ CC ライセンス（最新版は v4.0）

- クリエイティブ・コモンズで作成されたライセンス。
- 条文とは別に、条件をマークで示すことで分かりやすくしたライセンスで、マークにはクレジットの表示義務、営利目的での利用禁止、改変禁止、ライセンス継承義務の４種類があり、これらの組み合わせにより条件を示す。
- なお、本ライセンスは、OSI 承認のライセンスには含まれない。

主要なライセンスの条件

OSSライセンスの種類（例）	主要なライセンス条件					
	著作権表示義務	ライセンスの承継(同一条件で配布)	OSS(改変部分を含む)のソースコード提供義務	結合する他コードへの伝搬性(注 1)	特許権の行使制限等 (注 2)	保証免責責任制限
GPLv3	有	有	有	有	必須特許許諾。差別的ライセンス禁止	無保証で配布
GPLv2	有	有		有	明文なし	同上
LGPLv2.01	有	有	有	無(注３)	明文なし	同上
MPLv1.1, v2.0	有	有	有	無	ソース開示で権利不行使約束	同上
Apache V2.0	有	有	無	無	必須特許許諾。特許ライセンス終了条件	同上
修正版BSD	有	有	無	無	明文なし	同上
MIT	有	有	無	無	明文なし	同上

（注 1）「伝搬性」とは、ある OSS と一体化したソフトウェア全体（OSS の派生物）に対して、ソースコードの公開等、OSS ライセンス（許諾条件）が適用されること。
（注 2）「特許権の行使制限」には、①OSS の貢献者（作者）や配布者が OSS を配布する相手方（下流のユーザ）に対する「特許ライセンス義務」と、②OSS のライセンサーやコントリビュータに対して特許権を行使すると、自己に対する当該 OSS に実施されている特許ライセンスが自動的に消滅しまたは解除される「特許ライセンス終了条件」がある。
（注 3）LGPLv2.1 は、リンクによる自社開発プログラムへの伝搬性はないが、ユーザ自身の利用のための著作物の改変を許可し、その改変をデバッグするためのリバースエンジニアリングを許可しなければならない（LGPLv2.1 第 6 条）。数字のパラメタやデータ構造のレイアウト、アクセス機構または小さなマクロや小さなインライン関数(長さが 10行かそれ以下)のみ利用するならば、そのオブジェクトファイルの利用は制限されないとされている（LGPLv2.1 第 5 条）。

ユーザに提供すべき情報

■自社製品にOSSを組み込んだ場合、ユーザに提供すべき情報

- 利用している OSS の名称とバージョンの情報（対象の特定）
- OSS に添付されているライセンス条件と同一のライセンス文書
- 無保証であること（免責条項）の表示
- 配布する OSS の著作権表示（記載されていた著作権表示を削除しない）
- その他、個々の OSS のライセンス条件で定められた情報。
  （例）OSS を改変した場合には、改変の事実と改変者の名前
- 謝辞の記載
- GPL の場合は、ソースコードを提供すること等

#LinuC学習中

OSSコミュニティ

OSSコミュニティとは？

■OSSコミュニティとは？

- オープンソースソフトウェアの開発・改善、情報交換などを目的に、さまざまな立場の有志によって構成された仮想の組織。

#LinuC学習中

■コミュニティの種類

- **開発コミュニティ**
  - オープンソースを開発するコミュニティ
  - 企業がコミュニティを主導する場合もある

  （例）MySQL（Oracle）、JBoss（RedHat）など
- **ユーザーコミュニティ**
  - オープンソースを利用するにあたり、情報交換を行ったり、日本語ドキュメントの作成を行ったりするコミュニティ

  （例）日本MySQLユーザ会、日本PostgreSQLユーザ会など

■OSSコミュニティの変化

- ボランティア主導→企業主導へ。
- 大手企業が貢献を競争する場所

主なOSSコミュニティ

■主なOSSコミュニティ

#LinuC学習中

- Linux Foundation
  - Linux だけではなく、各種オープンソースコミュニティに対して、種々の施策を行って活動の支援をしている組織
  - 各種オープンソースプロジェクトの運営、コミュニティへの資金援助、インフラの提供、イベントの開催、トレーニングの提供などを実施。
- Apache Software Foundation
  - Web アプリケーションサーバー「Apache HTTP Server」をはじめとして、数多くのオープンソースプロジェクトを支援する非営利団体。
  - 代表的なプロジェクト：「Apache Tomcat」、「Log4j」、「JMeter」など。
- GNUプロジェクト
  - UNIX ライクな OS である「GNU」を開発するためにスタートしたプロジェクト。
  - OS 本体のほか、ユーティリティやアプリケーションなども開発。
  - Free Software Foundation (FSF)が支援。

OSSコミュニティに参加する方法

■OSSコミュニティに参加する方法

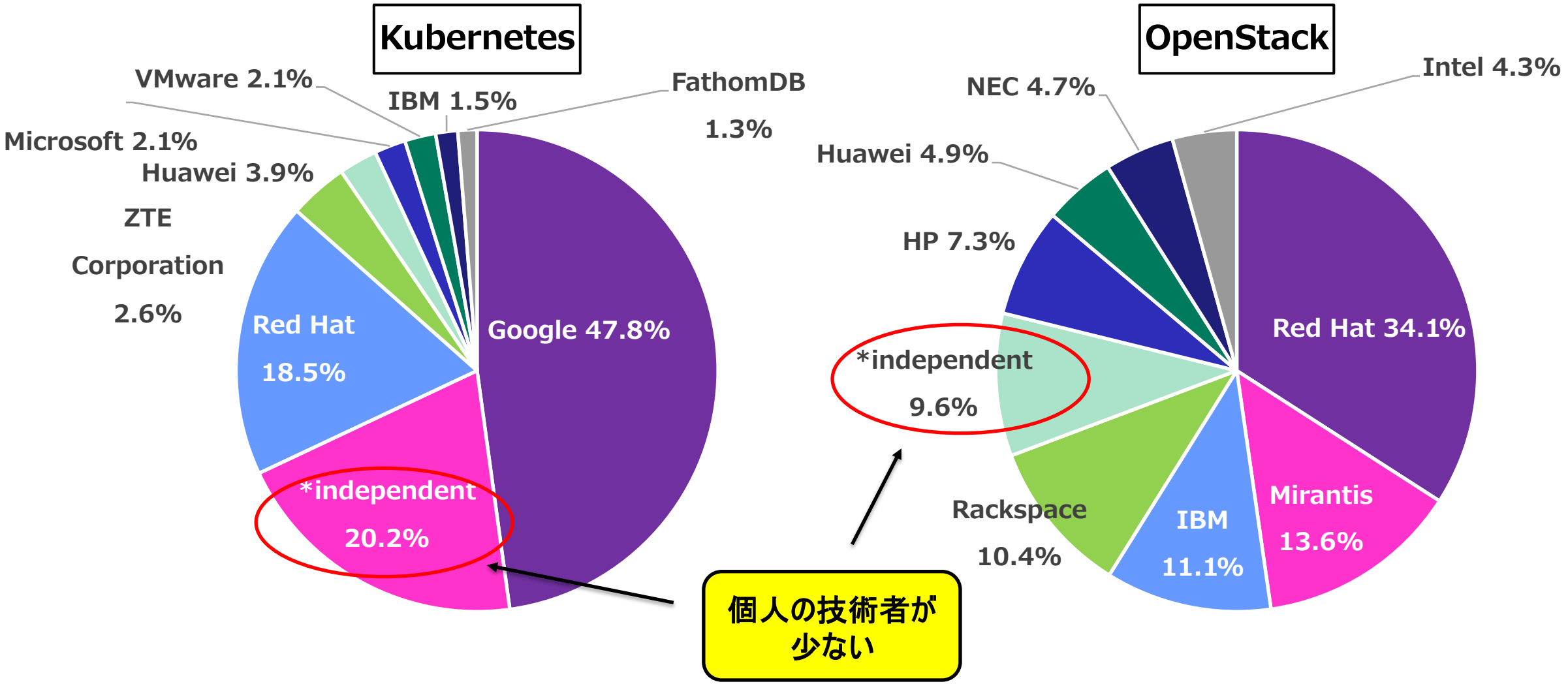
- コミュニティの活動内容
  - ① 開発（コア、拡張機能）
  - ② QA （バグレポート、テスト）
  - ③ L10N: ローカライゼーション（翻訳 / 言語ごとの機能開発）
  - ④ ドキュメント作成
  - ⑤ テンプレート作成
  - ⑥ マーケティング
  - ⑦ インフラ（ Web 、ビルドサーバー）
  - ⑧ ユーザーサポート（ Q&A サイト、 ML)
  - ⑨ イベント運営
  - ⑩ コミュニティ運営

#LinuC学習中

コードを書くことだけが貢献ではない！

誰が開発しているのか？①

■主なOSSコミュニティの開発者の状況



https://www.stackalytics.com/

誰が開発しているのか？②

◆ Linuxカーネルの場合

➢ Linux Foundationが発行している「Linux Kernel Development Report(*)」によると

- ✓ 「不明」と「なし」のグループを含めた上位 10 社が、カーネルに対する貢献の約 55%
- ✓ カーネル開発の 80% 以上は、企業の正規の仕事として行われている。
- ✓ 企業の支援を受けていない開発者からの貢献は、長期にわたって緩やかに減少傾向。
  2012年版：17.9%, 2013 年度版：13.6%, 2015 年度版：12.4%今回： 8.2%

2011		2012		2013		2015		2017	
社名	割合	企業名	割合	企業名	割合	企業名	割合	企業名	割合
なし	18.90%	なし	17.90%	なし	13.60%	なし	12.40%	Intel	13.10%
Red Hat	12.40%	Red Hat	11.90%	Red Hat	10.20%	Intel	10.50%	なし	8.20%
Novell	7.00%	Novell	6.40%	Intel	8.80%	Red Hat	8.40%	Red Hat	7.20%
IBM	6.90%	Intel	6.20%	Texas Instruments	4.10%	Linaro	5.60%	Linaro	5.60%
不明	6.40%	IBM	6.10%	Linaro	4.10%	Samsung	4.40%	不明	4.10%
Intel	5.80%	不明	5.10%	SUSE	3.50%	不明	4.00%	IBM	4.10%
consultants	2.60%	Consultant	3.00%	不明	3.30%	IBM	3.20%	consultants	3.30%
Oracle	2.30%	Oracle	2.10%	IBM	3.10%	SUSE	3.00%	Samsung	3.20%
Renesas Technology	1.40%	Academia	1.30%	Samsung	2.60%	Consultants	2.50%	SUSE	3.00%
The Linux Foundation	1.30%	Nokia	1.20%	Google	2.40%	Texas Instruments	2.40%	Google	3.00%
academics	1.30%	富士通	1.20%	Vision Engraving Systems	2.30%	Vision Engraving Systems	2.20%	AMD	2.70%
SGI	1.30%	Texas Instruments	1.10%	Consultants	1.70%	Google	2.10%	Renesas Electronics	2.00%
富士通	1.20%	Broadcom	1.10%	Wolfson Microelectronics	1.60%	Renesas Electronics	2.10%	Mellanox	2.00%

(*)https://go.pardot.com/l/6342/2017-10-24/3xr3f2/6342/188781/Publication_LinuxKernelReport_2017.pdf

なぜコントリビュートするのか？

■なぜコントリビュートするのか？

#LinuC学習中

➢個人的な理由

- ✓既に持っているスキルを改善する
- ✓似たようなことに興味を持っている人に出会う
- ✓メンターを見つけ互いに教え合う
- ✓あなたの評判（やキャリア）を育てるのに役立つ成果物を作り上げる
- ✓ピープルスキルを学ぶ
- ✓変化を起こせるようになる助けとなる、たとえそれが小さな変化であったとしても

➢ビジネス上の理由

- ✓優秀なOSS開発者の採用
- ✓メンテナンスコストの削減
- ✓プロジェクトの方向性への影響力

コントリビュートするプロジェクトの見つけ方①

■コントリビュートするプロジェクトを見つけ方

➢プロジェクトのチェックリスト①

- ✓オープンソースの定義に適っているか
- ✓コミットアクティビティのチェック
  - 最新のコミットはいつか？
  - そのプロジェクトには何人のコントリビューターがいるか？
  - どのくらいの頻度でコミットしているか？
- ✓プロジェクトのイシューへの対応状況
  - いくつのイシューがあるか？
  - メンテナーはイシューがオープンされたらすばやく反応しているか？
  - イシューでアクティブな議論は行われているか？
  - イシューは最近のものか？
  - イシューがクローズされているか？

#LinuC学習中

コントリビュートするプロジェクトの見つけ方②

■コントリビュートするプロジェクトを見つけ方②

#LinuC学習中

➢コントリビュートする前のチェックリスト②

✓プルリクエストへの対応状況

- いくつのプルリクエストがあるか？
- メンテナーはプルリクエストがオープンされたらすばやく反応しているか？
- プルリクエストでアクティブな議論は行われているか？
- プルリクエストは最近のものか？
- どのくらい最近プルリクエストがマージされたか？

✓プロジェクトは歓迎してくれるか

- メンテナーはイシューでの質問に助けとなるような回答をしているか？
- 人々はイシューやディスカッションフォーラム、チャットで友好的か?
- プルリクエストはレビューされているか？
- メンテナーはコントリビュートに対して感謝しているか？

良き企業市民になるために

■良き企業市民になるために

#LinuC学習中

- ➢まずコミュニティに参加する。
  - メーリングリスト、フォーラム、インターネット リレー チャット（IRC）、Slack、バグ トラッカー、ソースコード リポジトリなどが。
- ➢最初はリードオンリー ユーザーで。
  - コミュニティ文化を吸収するために相当な時間を費やし、このコミュニティの行動規範や期待される振る舞いを理解する。
- ➢ガバナンスを理解する。
  - プロジェクト内で、どのように意思決定が行われ、さまざまなコントリビューションに対して、誰が意思決定しているのかについて理解する。
- ➢小さく始める。
  - 簡単なバグやドキュメントの修正から始める。
- ➢イベント等で良い関係を構築する。
- ➢早めに、高頻度でコミュニティを巻き込む。
- ➢アップストリームにコントリビューションする。

コントリビューションするためのベストプラクティス

■コードをアップストリームにコントリビューションするためのベストプラクティス

#LinuC学習中

あなたの組織内部で推進すること

1. 正当な理由づけによって、アップストリームにコントリビューションする決定を下すこと。
2. アップストリームを念頭に置いてコードを設計し、実装すること。
3. 「上流第一（upstream first）」ポリシーを採用し、まずパッチをアップストリームに提出し、自社製品ではダウンストリームのものを採用すること。
4. たとえ大きな関与でなくても、あなたの開発者をオープンソース プロジェクトに関わり続けるようにすること。

外部との関わりで推進すること

1. あなたのコントリビューションが他の人にも役立つことを確認すること。
2. 適切なコーディング スタイルに従うこと。
3. プロジェクトのコントリビューション提出プロセスに従って作業すること。
4. あなたのコントリビューションについての説明やドキュメントを提供すること。
5. フィードバックに耳を傾けて、それに対応すること。
6. 辛抱強く、受け入れられるまでコードの改良、修正を続けること。

https://www.linuxfoundation.jp/resources/open-source-guides/participating-open-source-communities/

オープンソース コントリビューションを習得するための11項目の秘訣

■オープンソース コントリビューションを習得するための11項目の秘訣

#LinuC学習中

- オープンソース コントリビューションのためのポリシーとプロセスを確立する
- すべてのオープンソース コントリビューションの承認を監督するチームを設置する
- あなたのテクノロジーが活かせる領域に焦点を当てる
- コントリビューターに必要なITインフラとツールを提供する
- コントリビューションのためのベストプラクティスのトレーニングをスタッフに提供する
- コントリビューションをトラッキングし、効果を評価し、改善し、コミュニケーションをとる
- 経験の少ない開発者をトレーニングするための指導プログラムを確立する
- コントリビューション ガイドライン（どのようにすれば良いか、しても良いこと、してはいけないことなどを示す）を提供する
- 開発者がオープンソース関連で法務サポートを受けられるようにする
- あなたにとって最も価値のあるオープンソース コミュニティから雇用される
- 各プロジェクト コミュニティの持っている固有のプロセスやプラクティスに常に従う

https://www.linuxfoundation.jp/resources/open-source-guides/participating-open-source-communities/

OSSの開発スタイル

■OSSの開発スタイル

#LinuC学習中

- 伽藍とバザール
  - ソフトウェアの開発方式
  - 伽藍方式
    - 中央集権的に伽藍を組み上げるようなやり方。
    - 少数の高スキルの開発メンバーで慎重に開発。
    - 完成するまでベータ版すら出さないような、リリースに対して厳しい考え方。
  - バザール方式
    - Linuxの開発で採用されているやり方。
    - 任せられるものは他人に任せ、早めに頻繁にリリースする。
    - リリースに対しては厳しくない。

OSSコミュニティの構造①

■(一般的な)OSSコミュニティの構造

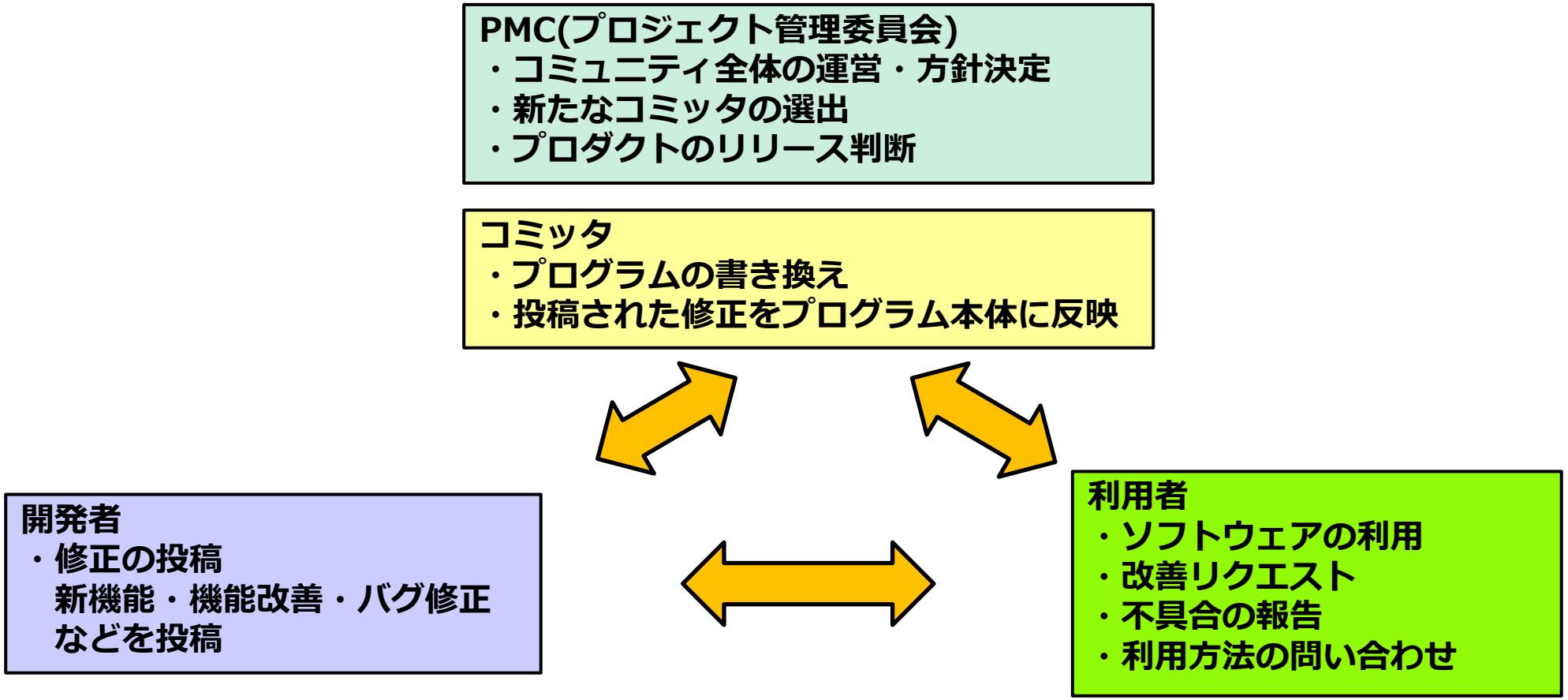
#LinuC学習中

#	役割	説明
1	リーダ	各コミュニティの調整を行う人物。企業内プロジェクトと異なり、一般的に民主的に選ばれ、民主的な運営を行う。
2	コミッタ（開発コミュニティのみ）	プログラムを変更する権限をもつ人物。コミュニティによって選出される。ソフトウェアの方向性について投票が行われる場合の投票権を持ちソフトウェアの方向性に関与できることが特徴。
3	活動メンバ（開発コミュニティではディベロッパともいう）	コミュニティに参加し、発言する、アクションを起こす人物。開発コミュニティでは、一部の機能の実装を行うこともある。
4	参加者	コミュニティに何らかの形で参加しているが、積極的なアクションを起こしていない、もしくは今後アクションをしようとしている人物。
5	支援者	コミュニティ運営に必要な資金、人材、物資などの提供を行う人もしくは団体。一般に、ソフトウェアの方向性に対する発言は行わずに支援する立場をとる。

OSSコミュニティの構造②

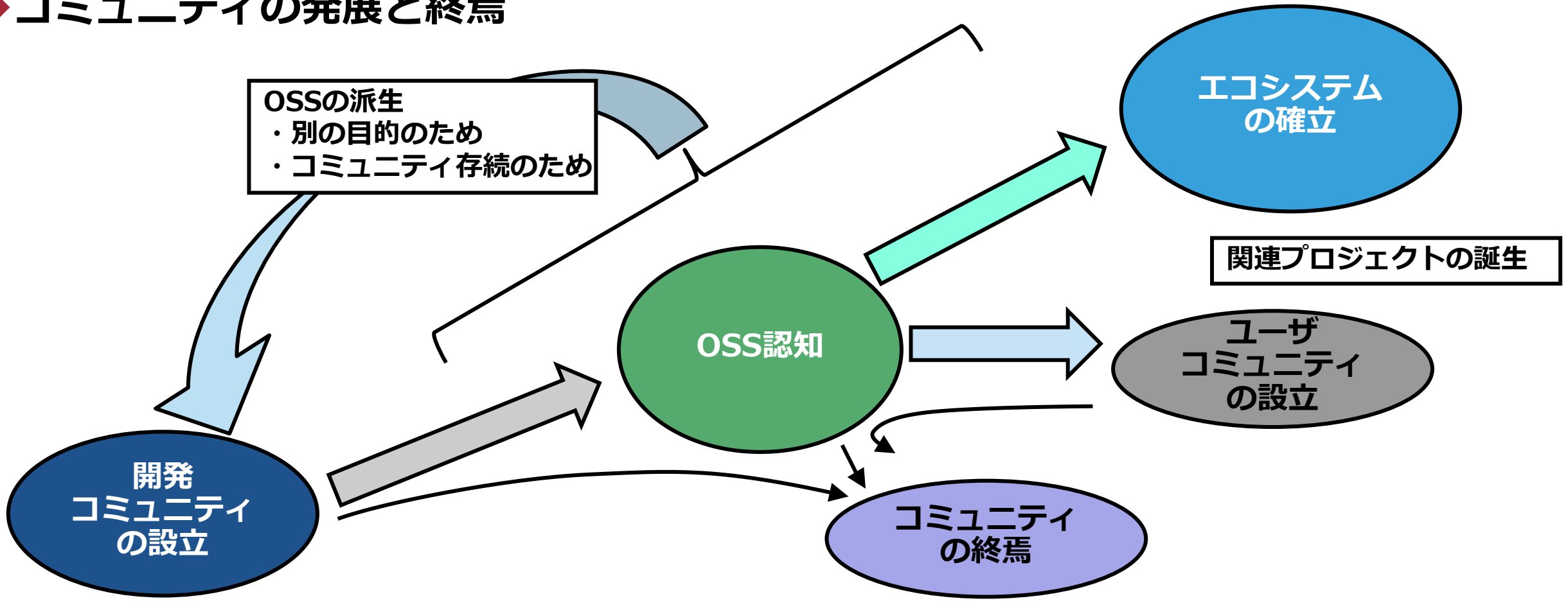
■ASFプロジェクトの構成
- トッププロジェクトの数：373プロジェクト
- コミッターの人数：7,826名

#LinuC学習中



コミュニティのライフサイクル①

◆コミュニティの発展と終焉



・少人数の開発者による形成
・企業のソフトウェア公開から形成
・既存OSSの派生から形成

・開発/保守など維持ができなくなる（資金不足）
・ユーザの支持を失う
・同じ役割を果たす新たなOSSの出現(世代交代)
・リーダの興味がなくなる

コミュニティのライフサイクル②

◆OpenOfficeの場合(誕生から・・・・)

1999：Sun Microsystems、StarVision買収

2000/10：「OpenOffice.org」プロジェクト立ち上げ

2010/01：Oracle、 Sun Microsystems買収 →プロジェクトの管理がOracleに移管

2010/05：一部メンバーが、「The Document Foundation」を立上げ
→「Libre Office」プロジェクト立ち上げ

エンジニアの流出
始まる

2011/04：Apache Software Foundationに移管
→「OpenOffice.org」終了。

2011/07：IBMが「Lotus Symphony」のソースコードを
「Apache OpenOffice」プロジェクトに寄贈。

2012/05：「Apache OpenOffice 3.4.0」リリース

「開発者不足」でプロジェクトが
存続するための条件を満たせない

2016/09：「Apache OpenOffice 」存続が困難に

2017/10：「Apache OpenOffice 4.1.4」リリース

OSSとビジネスモデル

ビジネスモデルについて

◆ビジネスモデル

➢ およそOSSのビジネスモデルは、下記の4つに分類できる。

#	モデル	内容	例
1	ディストリビューション モデル	自社またはコミュニティにて開発されたソフトウェアの配布とサポートを行うモデル (ex.Red Hat Enterprise Linux, PowerGres, MySQL, Miracle ZBX 等)	RedHat, SRA OSS, Oracle, Zabbix, サイバートラスト等
2	システム インテグレーションモデル	OSSを活用したシステム構築およびプロフェッショナルサービス（コンサルテーションを含む）を実施するモデル	NTTデータ, SIOS, SCSK,CTC等
3	サービスモデル	OSSを活用して構築したサービスを提供するモデル(ex. Amazon RDS 等)	AWS, Google,Facebook, 楽天,ヤフー,リクルート等
4	その他	ハードウェア販売などの目的達成のためにOSSを活用するモデル	ハードウェアベンダー (日立、富士通、NEC等) クラウドベンダ？

ビジネスモデル事例(コミッタの輩出)

◆NTTデータの場合

#LinuC学習中

■コミッタ就任

- 2014/12：Apache Hadoopおよびその関連のプロジェクトのコミッタに、小沢健史氏（NTTソフトウェアイノベーションセンタ）、鯵坂明氏、岩崎正剛氏（NTTデータ システム技術本部）の3人が就任。
- 2015/6：猿田浩輔氏(NTTデータ システム技術本部)が、分散データ処理ソフト「Spark」のコミッタに就任。
- 2016/1：鯵坂明氏と小沢健史氏は、Hadoopおよびその関連のプロジェクトのプロジェクトマネジメント委員(以下PMC)に就任。
- 2019/12：Apache AirflowのコミッターおよびPMCに関 堅吾氏が、Apache Ambariのコミッタに田中 正浩氏がそれぞれ就任。

■コミッタになるということ

- コミュニティとの連携により、高品質なサポートサービスが可能
- 社外への技術力のアピール

ビジネスモデル事例(OSS化の意義)

◆ミドクラの場合

#LinuC学習中

■ネットワーク仮想化ソフト「MidNet」をOSS化。(2014/11)

- 「MidoNet」とは？
  - ✓ オーバーレイ型のネットワーク仮想化を実現するソフトウェア。
  - ✓ 物理的なネットワーク構成の上に、仮想的なレイヤ2ネットワーク、レイヤ3ネットワークを構成し、ファイアウォール、ロードバランス、アクセスコントロールやセキュリティグループなどの機能を実現。
- 足掛け5年で25億円を要して開発したソフトウェア。

■なぜOSS化したか？

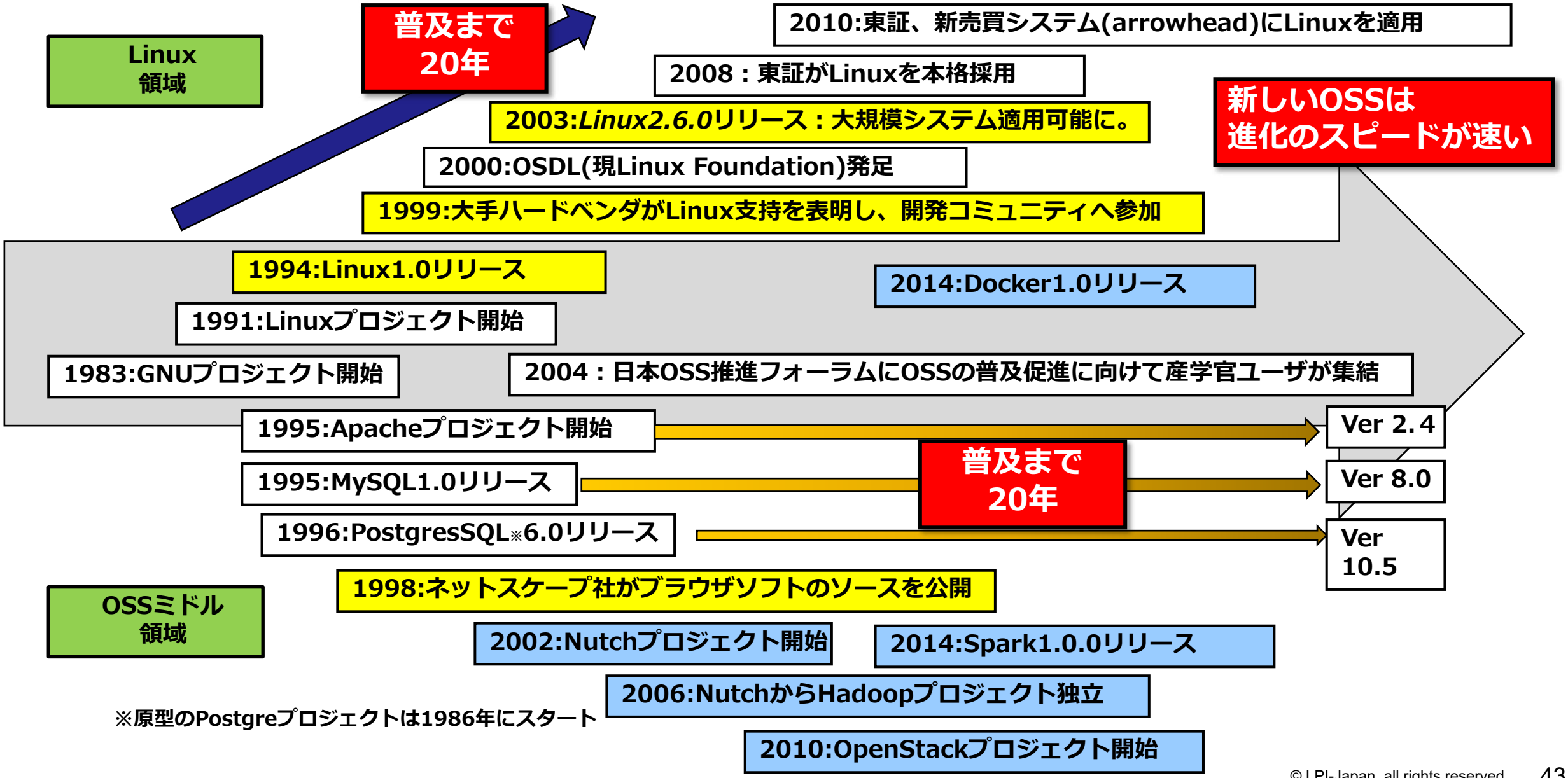
- 米国大手企業から、「ベンチャーが開発するソフトウェアは、OSSでなければ導入できない」と言われた。
- 開発ベンダーが買収されたり、開発が継続できなくなるようなことが発生し、継続的に利用することに支障が起きるようでは困る。

■OSS化の効果

- OSS化したことで、他社のエンジニアが検証した結果をブログ等で公開。
  → 認知度がそれまでに比べ飛躍的に向上。
- 技術力の高さの証明。

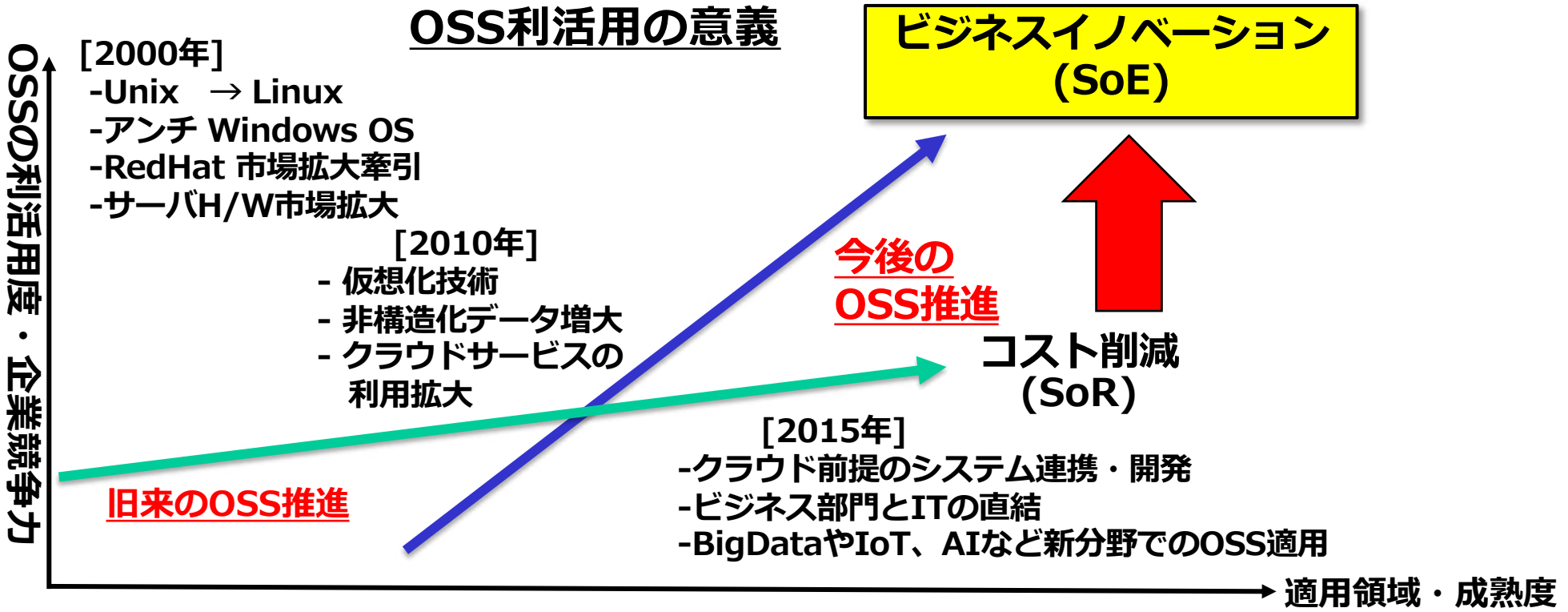
OSSの「これまで」と「これから」

オープンソースの『これまで』



※原型のPostgreプロジェクトは1986年にスタート

オープンソースの『これから』



主領域	OS領域	ミドル領域	アプリ・サービス領域
主導	情報システム部門	事業部門	企業体
目的	改善活動	IT化速度向上	市場創成、革新、企業競争力向上
企業間	コスト競争	協業・連携	エコシステム化
基盤	物理・仮想	単一クラウド	マルチクラウド、IoT、M2M

LinuC

ご清聴ありがとうございました

LPI-JAPAN