DevOpsエンジニアのための認定資格、DevOps Tools Engineerとは？

Linux Professional Institute 日本支部

岡田　賢治

発表の前に

'Ask the speaker'

終わりに10分程度の質疑応答の時間を設けます。

ぜひオンラインチャットで質問してください。

本日のサマリー

はじめに

試験概略

試験範囲説明

ソフトウエアエンジニアリング

まとめ

はじめに

本日のスピーカー

岡田賢治

10/1より都内某社のサラリーマンです。

普段は、SESで東京都内某所に通っております

最近もっぱら在宅（で体重増）

好きなディストリビューションはDebian

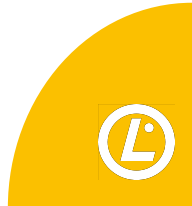
「LPI（日本支部）の”方から来ました”」

レビューアや試験開発手伝いなどを行なっています

試験概略

DevOpsとは1

- Develop/開発とOperation/運用を組み合わせてできた言葉
- 開発手法と運用手法を組み合わせて、柔軟かつスピーディーにシステムを開発すること
- サービスの開発・ビルド・テスト・リリース/デプロイが全て一体になったスキルが必要
- DevOpsに合わせ、開発方式やデプロイ方式が変更



DevOpsとは2

- ➔ FlickrでのDevOpsの話
  - ◆ DevOpsの紹介事例として、非常に広く言われた話。
  - ◆ Flickrでは、構成しているサーバを、”1日10回”壊して”いる」
    ⇨リリース/デプロイは誰しもナーバスになるのに、なぜに「定期的に”壊す”？」

- ➔ Flickrの主張
  - ◆ サーバインスタンスを構成する
  - ◆ 動作検証をきちんと行って、リリース/デプロイを行う。
  - ◆ 新しいインスタンスの追加時・古いインスタンスの削除時は、サービスが止まらないようなデプロイをしている。

LPIが考えるDevOps

➔ 以下の構成

- 概念・・・マイクロサービス・Immutable Server/Infrastructure・WebAPI
- ソースコード管理・・・Git
- CI/CD・・・デプロイ方法・Jenkins・テスト方法
- 環境構築・・・システムイメージ・インスタンス・コンテナ/
  Docker/Kubernetes・Ansible
- ログ収集・・・syslog, logstash, ElasticSearch/Kibana/Logstash(ELK)
  Prometheus(状態監視)

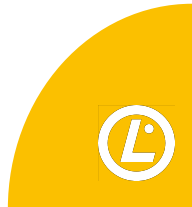
試験の概要

「DevOps環境を正しく運用するために必要な、実用的スキルに重点を置いています。また最もよく使用されるDevOpsツールの使用に必要なスキルに焦点を当てています。その結果、開発と運用の交点をカバーする認定資格が作成され、DevOpsの分野で働くすべてのITプロフェッショナルに関連するようになっています。」

- 試験のタイトルがDevOps "Tools"であること
- 受験に必要な資格や実務経験はない
  - 認定は、DevOps Tools Engineerの試験に合格すること。
- ピアソンビュー/Pearson Vueより配信
- 受験料: ¥15,000（税別）
- リテイクポリシー
  - 1回目のリテイクは、前の試験から1週間以上。
  - 2回目のリテイクは、前の試験から30日以上。
- 有意な認定期間は5年

このような方に最適の試験

- モダンなプログラミングツールのスキルを証明
  - Git, Jenkins
- モダンなインフラのスキルを証明
  - Docker, Vagrant, Swift, Ansible, Packer
- 概念の理解の証明
  - デプロイ, マイクロサービス, Web/RestAPI



試験範囲説明

試験範囲1

701 ソフトウエアエンジニアリング（総重量　合計18）

701.1 モダンなソフトウエア開発 (総重量: 6)

データの永続性・トランザクション・並行性・可用性・スケーリング・ロードバランス

701.2 ソフトウエアのコンポーネントとプラットホームの標準(総重量: 2)

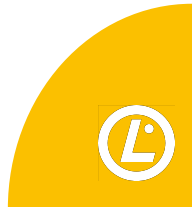
オブジェクトストレージ・SQL/NoSQL DB・CDN

701.3 ソースコード管理 (総重量: 5)

Git, Subversion, CVS

701.4 継続的インテグレーションと継続的デリバリ (総重量: 5)

CI/CD, 成果物/artifact, Jenkins, Blue-green/Canaryデプロイメント



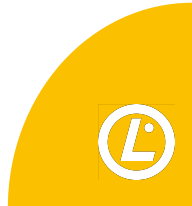
試験範囲2

702 コンテナ管理（総重量　合計 16）

702.1 コンテナの利用方法 (総重量: 7)
Docker, Dockerfile

702.2 コンテナのデプロイとオーケストレーション (総重量: 5)
docker-compose, docker, Docker Swarm, Kubernete/kubectl

702.3 コンテナインフラストラクチャー (総重量: 4)
Dockerネットワーク・Dockerストレージ・Dockerボリューム



試験範囲3

703 マシンデプロイメント（総重量　合計 8）

703.1 仮想マシンのデプロイメント (総重量: 4)

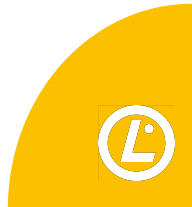
Vagrant, Vagrantfile

703.2 クラウドへのデプロイ (総重量: 2)

cloud-init

703.3 システムイメージの作成 (総重量: 2)

packer



試験範囲4

704 設定管理（総重量　合計 10）

704.1 Ansible (総重量: 8)

Ansible, Playbook

704.2 他の設定管理ツール (総重量: 2)

Puppet, Chef

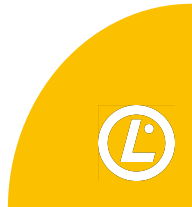
705 サービスオペレーション（総重量　合計 8）

705.1 IT オペレーションと監視 (総重量: 4)

Prometheus

705.2 ログの管理と分析 (総重量: 4)

logstash, Elasticsearch/Kibana



勉強方法と技術トピック

注目するべき試験課題

701 ソフトウエアエンジニアリング
  701.1 モダンなソフトウエア開発 (総重量: 6)
  701.3 ソースコード管理 (総重量: 5)

702 コンテナ管理（総重量　合計16）
  702.1 コンテナの利用方法(総重量: 7)
  702.2 コンテナのデプロイとオーケストレーション (総重量: 5)
  702.3 コンテナインフラストラクチャー (総重量: 4)

704 設定管理
  704.1 Ansible (総重量: 8)

上記6項目だけで、総重量合計36
全総重量は60。
60 x 0.6 = 36より、全体の60%をカバー

発表者からの私見を

あくまで私見ですので、LPI公式の見解ではない旨ご注意

- 正直な感想「いきなりは難しいと思う」
  - 最低でもLPIC-1、できればLPIC-2を持っていると、学習・試験合格が楽になる。
- 学習方法は、LPICの時と同じ「手を動かす」
- 「最近の方法」で学習して見てはどうでしょうか？
  - Qiitaなどのオンライン情報。
  - 発表者は、backlog.jpの「サルでもわかるGit」でGitの使い方を勉強しました。

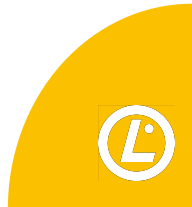
701 ソフトウエアエンジニアリング

701.3 ソースコード管理（総重量: 5 ）

- ソースコードはバージョン管理をする必要がある
- 複数人数での開発による、共同でのソースコード管理
- 一人での開発でも、更新履歴の管理等

- 古くはSCCS, RCSなどがUNIX界隈で使われていた。
- その後CVS, Subversionなどが使われるようになった（1990, 2000年代）

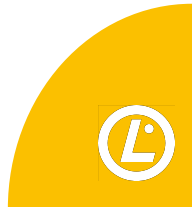
- 最近は、Gitが主流（2010年以降）



701 ソフトウエアエンジニアリング

Git

- 分散型バージョン管理システム
- 開発者は、Linuxのカーネル開発チーム
- それまで、Linuxのカーネルは、BitKeeperといわれるプロプライエタリのバージョン管理システムで管理されていた。
- バージョン管理は優秀なソフトだが、プロプライエタリは嫌だ。
- オープンソースにしてくれない・・・拒否
- そっくりなものを作った・・・のがGit



701 ソフトウエアエンジニアリング

Git

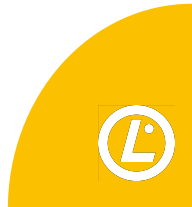
- 「分散型」バージョン管理システム
- リポジトリが、ローカルとリモートで「分散している」
- リポジトリ・・・ソースコード群の更新記録・分岐した情報
- 「リモートが更新されている」vs「ローカルが更新されている」
- その後、両者の同期をとる。

- 別のバージョン管理システムでは、
  - **のソースコードを変更
    ネットワークで転送・サーバ側では「ロック」＝他者が更新できない
  - ネットワークで転送するため、ネットワークに接続することが前提

701 ソフトウエアエンジニアリング

Gitでは

- ローカルに対して、更新を適用(Commit)。
- ローカルの更新なので、ネットワークに接続の必要はない。
- リモートの更新状況・ローカルをリモートに適用するときに、ネットワーク接続が必要。
- ローカルをリモートに適用(Push)
- すでにリモートが更新されていた・・・
- コンピュータが判断できるのであれば、更新がで適用される。
- 判断できなければ、コンフリクト状態。人手で操作する。



701 ソフトウエアエンジニアリング

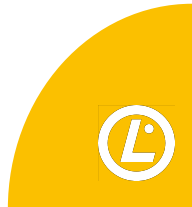
ブランチ

履歴の分岐

- ブランチは最初に1つ（masterという名称のことが多い）
- 分岐することで、ブランチを作成（ブランチを「切る」）
- ブランチを合流「マージ」

ブランチの切り方

- プログラマごとにブランチを作成。
- 結果を別ブランチにマージ・・・プログラマごとに作業を分散できる
- 機能ごとにブランチを作成
- マージしない限りその機能は反映されない・・・ブランチごとにビルド・テストを繰り返す。その間、本体には影響がない。
- 「パーティークラッカーじゃねぇ！」



701 ソフトウエアエンジニアリング

Gitを使ってみたい（試験範囲外）

サーバ

- 時々間違える人がいますが、「 Git + SSH (+Apache HTTPD/nginx)」だけでサーバは構成可能
- Gitの通信をSSHで行う、さらに通信の汎用性が高い HTTP(S)で通信

- ブランチのツリー（ネットワーク図）表示
- プルリクエスト作成
- wikiでのドキュメント作成
- チケット作成によるタスク管理 etc…

を便利に行うツールがある・・・ Web型Gitリポジトリマネージャ

701 ソフトウエアエンジニアリング

Web型Gitリポジトリマネージャ

- プロプライエタリ
  GitHub Enterprise(GitHub/MS), BitBucket(Atlassian), GitLab EE(GitLab)
- Saasで無償で利用可能
  github.com(GitHub/MS), bitbucket.org(Atlassian), GitLab.com(GitLab)
- オープンソース・無償
  GitLab CE(GitLab), gitbucket(@takezoe氏), gitblit, Gogs

Gitクライアント

「gitコマンド」・・・唯一無二の最強クライアント

SourceTree(Atlassian), Eclipse EGit-Plugin, GitKraken

まとめ

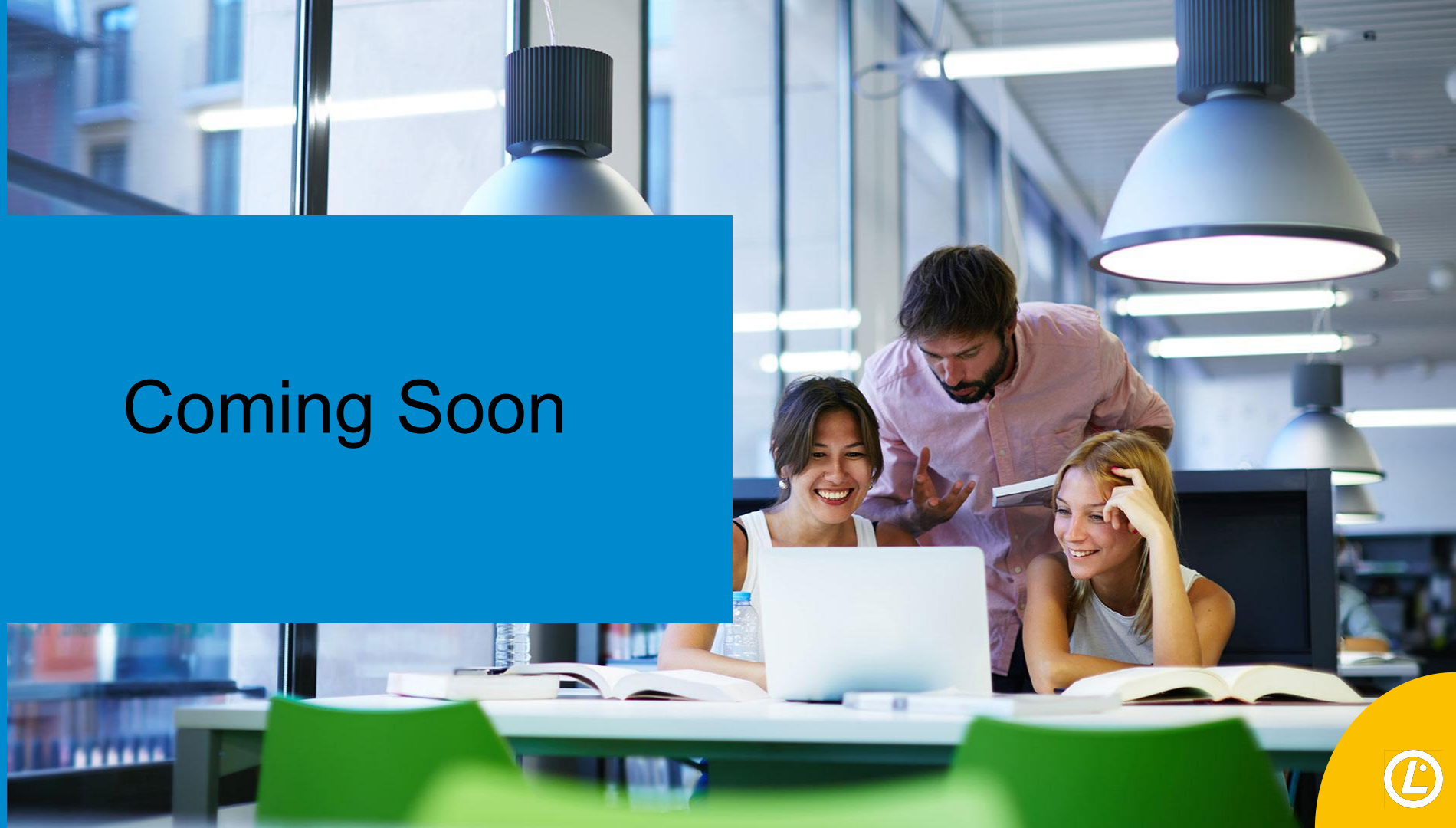
まとめ

- LPIが、DevOps Tools Engineer Examをリリースしました。
- Docker/コンテナ・Immutable Server/Infrastracture・Git・Ansible・・・話題のトピックばかり
- これらのスキルの習得・裏付けのために、チャレンジして見てください。
- 出版社からも問い合わせが来ています
- LPI・LPI日本支部からも、続々と情報をリリースします。
- ステッカー配布していますので、ぜひブースにもお立ち寄りください。

LPI日本支部はこのようなことをしています

ユーザコミュニティ設立

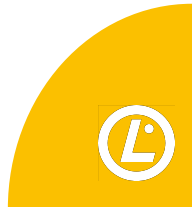
→「LPIC取った後のユーザはほったらかし。
　どうにかならない？」
→本国より「コミュニティ運動に注力したい」
→コミュニティ（＝LPIC資格ホルダー）の集まりを
2019/4/13に第一回＠東京を開催しました。
第一回関西として、本日この後開催！！
→活気溢れる意見交換
→いずれ(東)名阪福岡北海道に広げたいと考えてます。


Coming Soon

Coming Soon 1

BSD Specialist

- BSD認定試験
- OpenBSD, FreeBSD, NetBSDのスキル
- 英語版はすでにリリース
- 2020年夏/秋に日本語版リリース予定



Coming Soon 2

LPIC-3 大幅改訂/ バージョンアップ

300・・・Samba & FreeIPA
　　OpenLDAP廃止
303・・・セキュリティ
~~304・・・仮想化 & HA（現行）~~・・・廃止
305・・・仮想化 & コンテナ
306・・・HA & HA Storage

もちろん古いバージョンも並列配信（半年～1年）
https://wiki.lpi.org にて試験範囲を公開
2020年夏/秋リリース予定。

まとめ

コミュニティ活動も開始します。

Linux Professional Institute(LPI)は、LPI認定価値の維持を継続的に行い、認定者、これからの受験者、雇用者、LPIのパートナーにとって、価値ある国際認定として認識されるように努力して行きます。

Facebook、TwitterもFollowしてください！

https://www.lpi.org/ja/devops

ご静聴ありがとうございました