1，オブジェクト指向
「オブジェクト指向とは」

オブジェクト指向はプログラミングにかかわる概念のことです。オブジェクトの意味は「もの」や「物体」です。人が見たり手にしたりする「もの」がオブジェクトです。オブジェクト指向は、プログラミングにおいて開発するもの一つ一つを「もの」として捉えて、その仕組みを考えながら組み立てていくことです。オブジェクト指向の概念と仕組みを理解したうえで開発された「もの」は、それ自体が単独で動作できるので、より高度な機能を持った製品を作り出すことができます。そのため、プログラミングを使う現代のものづくりの現場などにとってオブジェクト指向は重要です。具体例として、車の免許を持っている人は、車の操作方法を知っており実際に運転することができます。しかし車の免許を持っていても車が動く仕組みを知っている人は少ないです。そのような車の構造や仕組みを理解することは車を運転するときに、必要はないからです。つまり、車という「もの」と車の操作という「役割」を切り分けて考えています。このような考え方がオブジェクト指向の考え方です。

参照
https://www.rstone-jp.com/column/200831no2/

2，GitHub flow
「GitHub flow とは」

GitHub Flowはブランチ戦略の1つで、GitHubで利用されているフローです。GitHub Flowの構成はシンプルでmainのmasterブランチと機能開発のためのfeatureブランチの2つのみです。特徴としては、Pull Requestを使うという点と、masterブランチは常にデプロイ可能という点があります。リモートリポジトリをチームで共有して開発する流れとして、まず、開発リポジトリを、自分のアカウントへforkします。masterブランチから、作業ブランチを切ります 。このときのブランチの名前は自由に決めて大丈夫です。次にローカルで開発、コミットし、Forkしたブランチへ Pushします。そして、本家リポジトリへPull Requestを出し、フィードバックを受けながら開発を進めます。最後にレビュアーに承認されたら、本家リポジトリへマージします。これでデプロイができます。

参照
https://qiita.com/tatane616/items/aec00cdc1b659761cf88

3，サーバーサイドエンジニア・フロントエンジニアの違い

フロントサイドエンジニアがWebのユーザーから見える部分の開発を行い、サーバー側のプログラムの実装やデータ処理などを行うのがサーバーサイドエンジニアです。フロントエンジニアの仕事は、ウェブデザイナーが作ったデザインをもとに画像ファイルなどを組み合わせながら、コードを記述しWebページを作成することです。また、サーバーサイドエンジニアの仕事内容は、サーバー側で実行する処理に必要なプログラムを開発することがメインの仕事内容です。Web上でユーザーが操作したときに操作内容に応じたプログラムの開発と保守をおこないます。なので、デザインに興味がある人はフロントエンドエンジニアを目指すことができ、スマホアプリやWebサービスのシステムに興味がある人はサーバーサイドエンジニアを目指すことができます。

参照

https://www.acrovision.jp/career/?p=2826

4，AWS
「AWSとは」

AWSは、Amazon社内のビジネス課題を解決するために生まれた IT インフラストラクチャのノウハウをもとに作られました。そして、アマゾン ウェブ サービスという名称でサービスがスタートしました。世界で人気があり数多く利用されているクラウドインフラサービスです。AWSの利用者は主に管理するエンジニアであり、AWSには主にサーバーまわりの最新資源の調達やパフォーマンスチューニング、そしてセキュリティ対策を任せることができ、人的リソースを費やす必要がなくなります。そのため利用者は自社サービスの開発や改善など、コアビジネスに集中することができます。AWSは、現在までさまざまな企業のウェブサイトやウェブサービスで利用されています。AWSは、ただ仮想サーバーを立てるだけでなく、他にもシステム構築や運用に必要なさまざまなサービスが提供されています。機能やメリットについて学ぶことで、ビジネスに活用することができます。

参照
https://business.ntt-east.co.jp/content/cloudsolution/column-37.html#section-02-02

5，Docker
「Dockerとは」

DockerはDocker社が開発したコンテナという概念を管理するソフトウェアです。サーバを起動する方法がシンプルで、かつ起動や処理が速いです。特徴は大きく分けて3つあります。まず、システム導入の高速化です。OSは既に共有して利用しているため、それらの設定は省き、システムを構築するために必要となる最低限のプログラムしかインストールする必要はありません。そのため、すぐにプログラムを適用し、導入までのスピードを高速化することができます。次の特徴として、起動時間の短縮化と多くの処理実装が可能です。完全仮想化であれば、サーバを起動する際にOSレベルから立ち上げていく必要があり時間がかかります。しかし、DockerではOSが既に立ち上がっているため、その分サーバーの起動時間を短縮化することができます。また、リソースの使用量が少なく済むためサーバへの負荷が低く、1度に多くのプログラム処理を実装することが可能です。最後の特徴として、コンテナ設定の再利用が可能です。1度作成したコンテナは、Dockerイメージを作成し、他のコンテナへ適用することにより再利用が可能となります。検証してみたい時や、リソースを拡大したい時など、すぐに同じ環境設定が適用されたコンテナを準備することができます。

参照
Dockerとは何かを入門者向けに解説！基本コマンドも｜Udemy メディア (benesse.co.jp)