【MFA(Multi-Factor Authentication)の設定】

AWSルートアカウントの乗っ取りを防ぐための2要素認証

<https://aws.amazon.com/jp/iam/details/mfa/>

Google Authenticatorを使用

AWSマネジメントコンソールのIAMからMFAの設定を行う。

一度サインアウトし、サイドログインすることで設定を確認。

　ログイン時にはGoogle Authenticatorを使用したログインも求められる。

【IAMユーザの設定】

AWSの操作を行う際はIAMユーザを使用する。AWSアカウントは権限が強く乗っ取られた場合にリカバリーすることができない。IAMユーザはユーザごとに細かい権限管理が可能な上に、乗っ取られた場合でもAAWSアカウントで削除が可能。

sysmgr/1129@gab

アクセスキーID：AKIAIXMMCZB7V2OXC6VA

シークレットアクセスキー：wmASg9h9uc/ELbGrKzy0g22v0nrdBVbXZIX7x4+B

ログインURL

<https://540381411801.signin.aws.amazon.com/console>

変更↓

https://teruya-admin.signin.aws.amazon.com/console

アカウント：teruya-admin

ID：terujun44@gmail.com

パス：@gab

【IAMポリシーの設定】

ダッシュボードのポリシーから設定。

詳細は割愛。

JSONでルールを記述するため、下記URLを参照

・IAM　JSONのポリシーエレメントのリファレンス

http://docs.aws.amazon.com/ja\_jp/IAM/latest/UserGuide/AccessPolicyLanguage\_ElementDescriptions.html

基本的には、カスタマー運用ポリシーを使用するとよい。

AES管理ポリシー自動で作成、更新されるがAWS管理のためこちらの意図しないタイミング、内容での変更の可能性もある。

【ポリシー付与】

基本的にはIAMユーザごとではなく、グループごとにポリシーを付与する。

【AWSCLI】

AWSコマンドラインインターフェース

<http://aws.amazon.com/jp/cli>

AWSマネジメントコンソールと同等の操作をコマンドラインで操作できる。

様々なOSに対応したソフトウェアがあるため、OSに適したインスーラを使用する。

・mac

Sudo brew install python

Sudo easy\_install pip

Sudo pip install awscli

バージョン確認

Aws –version

アップデートは以下

Pip install –U awscli

・Linux(CentOS7)

Sudo yum install python-setuptools

Sudo easy\_install pip

Sudo pip install awscli

バージョン確認

Aws –version

アップデート

Pip install –U awscli

・Windows

64ビット

<https://s3.amazonaws.com/aws-cli/AWCLI64.msi>

32ビット

<https://s3.amazonaws.com/aws-cli/AWCLI32.msi>

バージョン確認

Aws --version

実際のコマンド

C:\Windows\system32>aws configure

AWS Access Key ID [None]: AKIAIXMMCZB7V2OXC6VA

AWS Secret Access Key [None]: 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

Default region name [None]: ap-northeast-1

Default output format [None]: text

C:\Windows\system32>aws ec2 describe-security-groups

SECURITYGROUPS default VPC security group sg-96b104ee default 540381411801 vpc-26d4bc41

IPPERMISSIONS -1

USERIDGROUPPAIRS sg-96b104ee 540381411801

IPPERMISSIONSEGRESS -1

IPRANGES 0.0.0.0/0

【CLI基本的な使用方法】

$ aws ＜サービス名＞＜リソース操作コマンド＞

例

$ aws ec2 describe-security-groups

$ aws s3 ls

基本オプション

--profile 設定したprofileを指定してコマンドを実行する

--region　リージョンの指定

--output　出力形式の指定

--filters　参照系のコマンド使用時に検索条件を指定してフィルタリングする

--query　実行結果内容を絞り込んで出力する

--region　/ --output　一時的にデフォルト以外を指定できる

--filtersオプションによる検索条件指定

参照系コマンド実行時に検索条件を指定することができる

<http://docs.aws.amazon.com/cli/latest>

aws ＜サービス名＞＜リソース操作コマンド＞ –filters “Name=＜フィルタ名A＞,Values=＜条件A1＞” ”Name=＜フィルター名B＞,Values=＜条件B1＞,＜条件B2＞”・・・・・

例

1.10.0.0.10のプライベートIPを持つインスタンス参照する場合

Ec2⇒describe-instanes

$ aws ec2 describe-instances –filters “Name=private-ip-address,Values=10.0.0.10

--queryオプションにゆおるい結果出力の絞込み

<http://docs.aws.amazon.com/cli/latest>

aws ＜サービス名＞＜リソース操作コマンド＞ –query ‘＜クエリー名(1階層目)＞’

例

全てのインスタンスに対してインスタンスIDのみを参照する場合

Aws ec2 describe-instances –query ‘Reservations[].Instances[].InstandeId’

【VPCネットワーク】

設定項目

1.VPC ネットワークアドレス

2.サブネット　　各AZに1つずつ。/20のネットワークアドレス

3.ルートテーブル　　VPC内のネットワークアドレス範囲内のの通信はVPC内部へルーティング。それ以外はインターネットへ向けた通信

4.インターネットゲートウェイ　　インターネットへの接続用ゲートウェイ

5.ネットワークACL　　　インバウンドとアウトバウンド両方の通信設定

6.セキュリティグループ　　通信セキュリティ

AWS

AWS

④インターネットゲートウェイ

VPC

172.16.0.0.

172.16.1.0.

172.16.2.0.

③ルートテーブル

⑤ネットワークACL

⑥セキュリティグループ

②サブネット　AZ-a

②サブネットAZ-b

1. VPC

リージョン

CLIでVPCを作成

Aws ec2 createvpc- --cidr-block 10.0.0.0/16

Aws ec2 modify-vpc-attribute –vpc-id vpc-0f96e859b9873183c --enable-dns-hostnames

CLIでサブネットを作成

Aws ec2 create-subnet –vpc-id vcp-0f96e859b9873183c –availability-zone ap-northeast-1a –cidr-block 10.0.0.0/24

CLIでルートテーブルを作成

Aws ec2 criate-route-table –vpc-id vpc-0f96e859b9873183c

CLIで関連付けを行う

Aws ec2 associate-route-table –route-table-id rtb-0c624bb2fb432dbf3 –subnet-id subnet-0560bceaf0f333f05

CLIでインターネットゲートウェイ

aws ec2 attach-internet-gateway --vpc-id "vpc-0f96e859b9873183c" --internet-gateway-id "igw-045dd116c8ed75d7e" --region ap-northeast-1

CLIでルートテーブルと関連付ける

Aws ec2 ceate-route –route-table-id rtb-0c624bb2fb432dbf3 –destination-cidr-block 0.0.0.0/0 –gateway-id igw-045dd116c8ed75d7e

【AWS操作用の公開鍵、秘密鍵の作成】

キーペアを作成し、SSHやRDPでアクセスする。

EC2⇒キーペア⇒キーペアの作成

CLIでキーペアを作成する

Aws ec2 create-key-pair –key-name myKeyPair –query ‘KeyMaterial’ –output text > myKeyPair.pem

公開鍵をインポートすることも可能

今回は省略。

【セキュリティグループの作成】

セキュリティグループ＝FW(ホワイトリスト)

EC2⇒セキュリティグループ⇒セキュリティグループの作成

管理者用のSSH通信を許可

CLIからの設定

セキュリティグループの作成

Aws ec2 create-security-group –group-name mySecurityGroup –description “my First Security Group” –vpc-id vpc-0f96e859b9873183c

下記のコマンドで現在アクセス元となっている自分のIPを確認する。

Curl http://checkip.amazonaws.com/

セキュリティグループのインバウンドに設定

Aws ec2 authorize-security-group-ingress –group-id sg-0ee3f903da9be2329 –protocol tcp –port 22 –cidr 153.220.204.79/32

設定の確認コマンド

Aws ec2 describe-security-groups –group-ids sg-0ee3f903da9be2329 –output json

【EC2インスタンスの作成】

EC2⇒インスタンス⇒インスタンスの作成

高度な詳細について

インスタンス起動時に実行するコマンドを記述できる。

例 シェルスクリプト

#! /bin/bash

Yum install httpd –y

例　cloud-initディレクティブ

#cloud-confi g

Packages:

* Httpd

色々出力コマンド

C:\Users\j-teruya>aws ec2 run-instances --image-id ami-92df37ed --count 1 --instance-type t2.micro --key-name blog\_20180527\_Key --security-group-ids sg-0ee

3f903da9be2329 --subnet-id subnet-0560bceaf0f333f05

【CLIからAMIを作成する】

Aws ec2 create-image –instance-id i-006cf3b8c74ca9891 –name “myFirstEC2” –description “myFirstEC2-AMI”

{

“ImageId”: “ami-dc3da3b4”

}

【EIPの利用】

静的に割り当てられるIPのこと

EC2⇒Elastic IP⇒新しいアドレスの割り当て⇒アクション⇒アドレスの関連付

54.64.138.154

【ELBの利用】

アクセスを複数台のEC2インスタンスに振り分ける

・CLB…第一世代

・ALB…第二世代

ELBはアクセス負荷に応じて自動でリソースの拡張を行う。

しかし、瞬間的に負荷が増えた場合にはリソースの自動拡張が追いつかない場合がある。

その場合はAMAZONへの事前申請にて可能。Pre-Warming

ELBの拡張・縮小はスケールアップ・ダウン　スケールアウト・インの２方式が状況に応じて行われえる。どちらの場合もELBへのアクセスIPが変更されるため、ELBにアクセスする場合は必ずELB作成時に生成されるドメイン名DNSでアクセスする

Cross-Zone Load Balancing

複数AZのEC2インスタンス全てに対してリクエストの割り振りが可能

AZ内のEC2の数に左右されない、均等な分散が可能になるのでONが推奨

External-ELB

インターネットからのリクエストを受け付ける

Internal-ELB

　VCP内のリクエストのみ受け付ける

ヘルスチェック

ポート監視　ポートのListen状態を確認

サービス監視 　HTTPでのHTMLへのアクセス可否を確認する

SSLターミネーション

　SSL証明書を認証する機能。SSL証明書をELBで管理できる

スティッキーセッション

　ＥＬＢが独自に発行するCookieを利用する方式とアプリケーションが発行したCookieを利用する方式の２種類が選択できる。

しかし最近は各インスタンス間でセッション情報を共有する仕組みがある。

⇒Memcached やRedis,　　AWSのElastiCacheというサービス

上記を使用することが推奨！！！！

ログ取得機能

　ＥＬＢで処理したリクエストのログをＳ3に保存する

CLIでの作成

Aws elb create-load-balancer –load-balancername External-Web –listeners Protocol=HTTP,LoadBalancerPort=80,InstanceProtocol=HTTP,InstancePort=80 –tag Key=NAMW,Value=External-Web –subnets 設定するsubnet –security-groups ELB用のセキュリティグループ

ＥＣ２の割り当て

Aws elb register-instances-with-load-balancer –load-balancer-name External-Web –instances ec2インスタンスのID ec2インスタンスのID

【WordPressを使ったブログサイトの構築】

AMIMOTO　株式会社デジタルキューブが運営しているマーケットプレイス

AWS

AWS

VPC

AWS

④インターネットゲートウェイ

W

パブリックサブネット

Ｍ

Ｓ

MySQL マスター

MySQL　スレーブ

プライベートサブネット

プライベートサブネット

・カスタムＶＰＣの作成

VPC⇒VPC⇒VPCの作成

設定例

　WP-PublicSubnet-A 10.1.11.0/24

　WP-PrivateSubnet-A 10.1.15.0/24

　WP-PublicSubnet-C 10.1.51.0/24

　WP-PrivateSubnet-C 10.1.55.0/24

・インターネットゲートウェイの作成

VPC⇒インターネットゲートウェイ⇒インターネットゲートウェイの作成

・ルートテーブルの作成

VPC⇒ルートテーブル⇒ルートテーブルの作成

・RDSの作成

ＤＢサブネットグループの作成

RDS⇒サブネットグループ⇒DBサブネットグループの作成

RDB⇒インスタンス⇒DBインスタンス

DBインスタンス識別子　wp-mysql

マスターユーザの名前　　root

マスターパスワード　　　1129jabjab

【最小構成のwebサーバの作成】

VPC

　Vpc-WordPress

　Vpc-id ：vpc-0b6e8ad0b3430b0a5

　IPv4：10.1.0.0/16

ルートテーブル：[rtb-001bc834515e8cfb8](https://ap-northeast-1.console.aws.amazon.com/vpc/home?region=ap-northeast-1#routetables:filter=rtb-001bc834515e8cfb8)

サブネット

**パブリックサブネット**

サブネットID：subnet-0b946fa02db4d156e

　IPv4：10.1.1.0/24

アベーラビリティーティーゾーン：ap-northeast-1d

**プライベートサブネット**

1. サブネットID：subnet-0b946fa02db4d156e

　IPv4：10.1.3.0/24

アベーラビリティーティーゾーン：ap-northeast-1d

1. サブネットID：:subnet-086de3c1f0989bd75

　IPv4：10.1.13.0/24

アベーラビリティーティーゾーン：ap-northeast-1a

セキュリティグループ

　①AP-Web-DMZ

グループID：sg-0fb7294c188ba4976

　②WP-DB

　グループID：sg-0acb39c39ef333eb0

DB用サブネットグループ

　サブネットグループ名：

　VPCID ：vpc-WordPress (vpc-0b6e8ad0b3430b0a5)

ARN：

DBの構築

　DBインスタンス識別子：wp-mysql2

マスターユーザの名前：root

マスターパスワード：1129jabjab

エンドポイント：wp-mysql2.c8gxmbddzpgn.ap-northeast-1.rds.amazonaws.com

【DB環境準備】

EC2⇒RDS

Mysql –u root –p –h wp-mysql3.c8gxmbddzpgn.ap-northeast-1.rds.amazonaws.com

Create user ‘wp-user’@’%’ identified by ‘wp-jabjab’;

Create database wordpress;

Grant all privileges on wordpress.\* to “wp-user”@”%”;

Flush privileges;

【WEBサーバの構築】

省略

【Webサーバカスタマイズ】

　sudo yum –y update

sudo yum –y install httpd mysql php php-mysql php-mbstring php-gd

sudo /sbin/chkconfig httpd on

sudo /etc/init.d/httpd start

wget [http://ja.wordpress.org/latest-ja.tar.gz ~/](http://ja.wordpress.org/latest-ja.tar.gz%20~/)

tar zxvf ~/latest-ja.tar.gz

sudo cp –r ~/wordpress/\* /var/www/html

sudo chown apache:apache –R /var/www/html

【構築サイト】

設定用URL：http:// 54.64.138.154/wp-admin/setup-config.php

サイトのタイトル：SENS

ユーザ名：teru\_goto

パスワード：jun\_teruki

ログイン用URL：<http://54.64.138.154/wp-admin/>

サイトアクセス用URL：http://54.64.138.154/index.php

Akismet

APIキー：cae320188a34

【FTPの準備】

参考URL

https://qiita.com/Yuki\_BB3/items/deeb84c360c6f3c37f51

ftpconfig.shとして下記を作成し実行する。

#!/bin/bash

sudo yum install vsftpd -y

sudo cp /etc/vsftpd/vsftpd.conf /etc/vsftpd/vsftpd.conf.org

Elastic\_IP=`curl http://169.254.169.254/latest/meta-data/public-ipv4`

sudo sh -c "cat << EOF >> /etc/vsftpd/vsftpd.conf

pasv\_enable=YES

pasv\_addr\_resolve=YES

pasv\_address=${Elastic\_IP}

pasv\_min\_port=60001

pasv\_max\_port=60010

use\_localtime=YES

force\_dot\_files=YES

EOF"

sudo sed -i -e "/^anonymous\_enable/s/YES/NO/g" /etc/vsftpd/vsftpd.conf

sudo sed -i -e "/^dirmessage\_enable/s/YES/NO/g" /etc/vsftpd/vsftpd.conf

sudo sed -i -e "/^#ascii\_upload\_enable/s/#//g" /etc/vsftpd/vsftpd.conf

sudo sed -i -e "/^#ascii\_download\_enable/s/#//g" /etc/vsftpd/vsftpd.conf

sudo sed -i -e "/^#chroot\_local\_user/s/#//g" /etc/vsftpd/vsftpd.conf

sudo sed -i -e "/^#chroot\_list\_enable/s/#//g" /etc/vsftpd/vsftpd.conf

sudo sed -i -e "/^tcp\_wrappers/s/YES/NO/g" /etc/vsftpd/vsftpd.conf

sudo sed -i -e "/^connect\_from\_port\_20/s/YES/NO/g" /etc/vsftpd/vsftpd.conf

sudo sed -i -e "/^xferlog\_std\_format/s/YES/NO/g" /etc/vsftpd/vsftpd.conf

sudo touch /etc/vsftpd/chroot\_list

sudo service vsftpd start

sudo chkconfig vsftpd on

セキュリティグループに下記を許可する

インバウンド

Tcp 21

Tcp 60001-60010

※下記はDB内書き換え用のメモ

http://kawatama.net/web/1638

【S3との連携】

S3とは

　ストレージとしての機能と静的Webホスティングの機能を兼ね備えたなんかよくわからんもの。

作成

　サービス⇒S3⇒バケットを作成する

バケット名：wp-backet-one

リージョン：アジアパシフィック (東京)

それ以外はデフォルト値

Wordpressとの連携

<http://chilp.it/f9b7c1>からプラグインをダウンロード

wordpressの管理者画面からプラグイン⇒新規追加⇒プラグインのアップロード

⇒ファイルを選択⇒今すぐインストール⇒プラグインを有効化⇒設定

アクセスキーとシークレットアクセスキーを入力⇒authebnticate

以下を設定

　Use this backet ：作成したバケット

　File uploads：チェックを入れる

　File permission：チェックを入れる

【ELB】※今回は使用していない

EC2⇒ロードバランサー⇒ロードバランサーの作成

ロードバランサーの設定

　名前：任意

スキーマ：インターネット向け

IPアドレスタイプ：ipv4

　ロードバランサーのプロトコル：http 80

　VPC

　サブネット：２つのフロントサブネットを追加

ELB用のセキュリティグループの作成が必要

　HTTPを許可

ルーティングの設定

　新しいターゲットグループ

　2つのターゲットを登録

Wordpressの変更も必要

　Wordpressの管理画面ログイン⇒設定⇒Wordpressアドレス（URL）とサイトアドレス(URL)をELBのDNSに合わせて変更し、画面下の変更を保存をクリック。

スティッキーセッション

　ELBからEC2へアクセスを振り分ける際のセッション情報使用設定

　EC2⇒ターゲットグループ⇒ターゲットグループを選択し説明タブの属性の編集⇒有効化にチェック

HTTPSでのアクセス

　自己証明書を作成することで簡単にhttpsでの通信が可能

　openssl genrsa –out ./server.key 2048

　　※パスフレーズ設定ありの鍵はELBでは使用できない

　CSRを作成

　openssl req –new –key ./server.key –out ./server.csr

対話形式で項目を入力。Common nameはELBのDNS名！

　CRTを作成する

　opennssl x509 –in server.csr –days 365 –req –signkey server.key –out server.crt

・server.crt

　・server.csr

　・server.key

上記が作成される

ELBの設定

　EC2⇒ロードバランサー⇒wp-slbを選択⇒リスナー⇒リスナーの追加

　プロトコルにHTTPSを追加

　ポート：443

　証明書をACMにアップロードするを選択

　プライベートキー、パブリック証明書に先ほど作成した自己証明書の内容を入力して作成をクリック

Wordpressの設定変更

　プラグイン⇒新規追加⇒SSLと入力して検索

【AWS Elastic Beanstalkを用いた動的サイト】

実はこれまでに作成したEC2+RDSの組み合わせのサイトは数クリックでつくれる

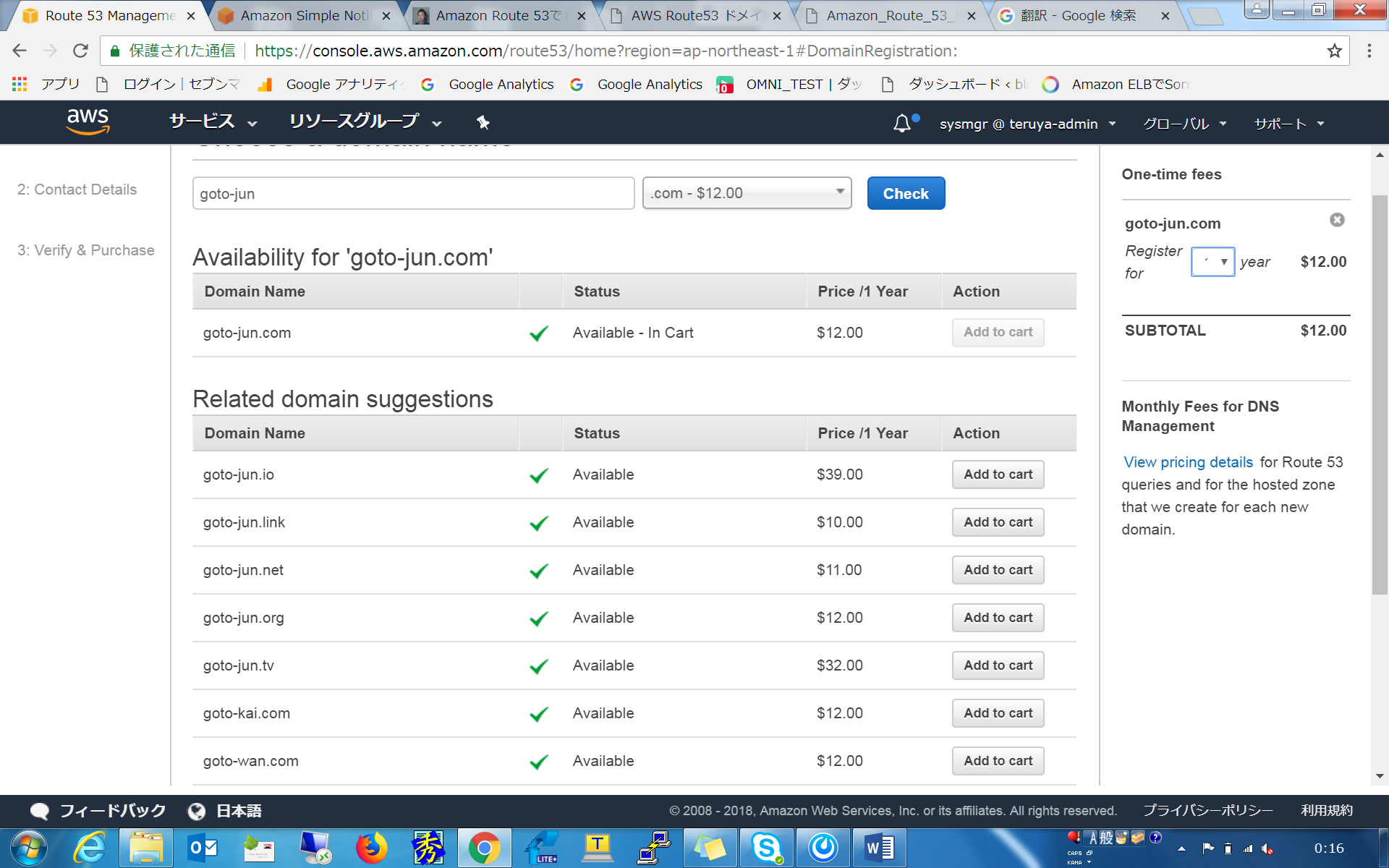
詳細は割愛

【route53】

ドメイン名の取得

サービス⇒Route53⇒Domain Registration　Get started now

Registar Domainをクリック



取得ドメイン

　goto-jun.com

　年間費用12ドル

【ワードプレス】

【Googleアカウント】

ID：[gototeruya@gmail.com](mailto:gototeruya@gmail.com)

パスワード：jun-teruki

【GITについて】

ID：junteruki

メールアドレス：[gototeruya@gmail.com](mailto:gototeruya@gmail.com)

パスワード：jun5teruki

★初期準備

1. 作業用ディレクトリを任意の場所に作成
2. 作業用ディレクトリ上で初期リポジトリを作成  
   git init
3. 共有リポジトリからクローン  
   git clone https://github.com/junteruki/Muscle-training-SNS/
4. ファイル索引追加(コミット対象にする)  
   git add <filename>
5. コミット  
   git commit –m “コメント”
6. 共有リポジトリにプッシュする  
   git push origin
7. (やんなくてもいい)共有リポジトリからクローンしていない場合は共有リポジトリを登録する必要がある  
   git remote add origin <サーバ>
8. 開発前には共有リポジトリに管理されている最新版のソースを取得する  
   git pull

★基本的な開発の流れ

マスタブランチで開発を推進していく場合には★初期準備の④と⑤を繰り返し、⑥でプッシュすればよい

★マスタ以外のブランチを作る場合

例）マスタブランチで開発をしているが、過去リリースしたコミット済のある機能について、性能向上が期待できるロジックを思いついた。しかし、実際に試してみないとわからないため試験的に試すことになった。間違っても共有リポジトリに反映させることは検証が終わってからとなる。

上記のようにマスタブランチの開発とは関係ない機能を試す場合に利用するのが一般的なブランチの利用方法