1. Introduction

We have three environments on AWS (wadzpay-aws-dev, wadzpay-aws-test, wadzpay-aws-prod). On each of them, there are two EC2 instances deployed: bitgo and bastion which you can connect, restart, stop, etc. In Cloud Watch, under log groups, you can see wadzpay and bitgo logs and see the last times, when applications were redeployed. In Elastic Container Service (ECS), under “task definition” you can see ENV variables with which applications are started and in Parameter Store you can see secret variables. You can edit task definition JSON to change variables and restart the app, though it is not recommended as any Terraform changes will rewrite the variables.

2. How to deploy

Now deployment is done automatically once you merge something in dev, test, or master. So, you only have to merge the feature branch into one of these branches and the target environment will be redeployed. For production, redeployment requires additional confirmation. Our company will stop working on this project soon. You will have to set up CircleCI or another CI again in your new repository to deploy changes to AWS. CircleCI is recommended way because the necessary config is in the “.circleci” folder.

3. How to change variables

Basically, you need files with secret variables put into the `terraform-aws/environments` folder. For these files, please contact Danylo. If you want to specify a secret variable, you need to set it to those files (based on the environment of course) If it’s a non-secret variable, feel free to set it into `{env}.tfvars.` First, you need to select the environment, you will be working with:

```
terraform init --backend-config="bucket=wadzpay-tf-state-{env}"
--backend-config="key=terraform/wadzpay-{env}" --backend-config="region={region}"
-reconfigure
```

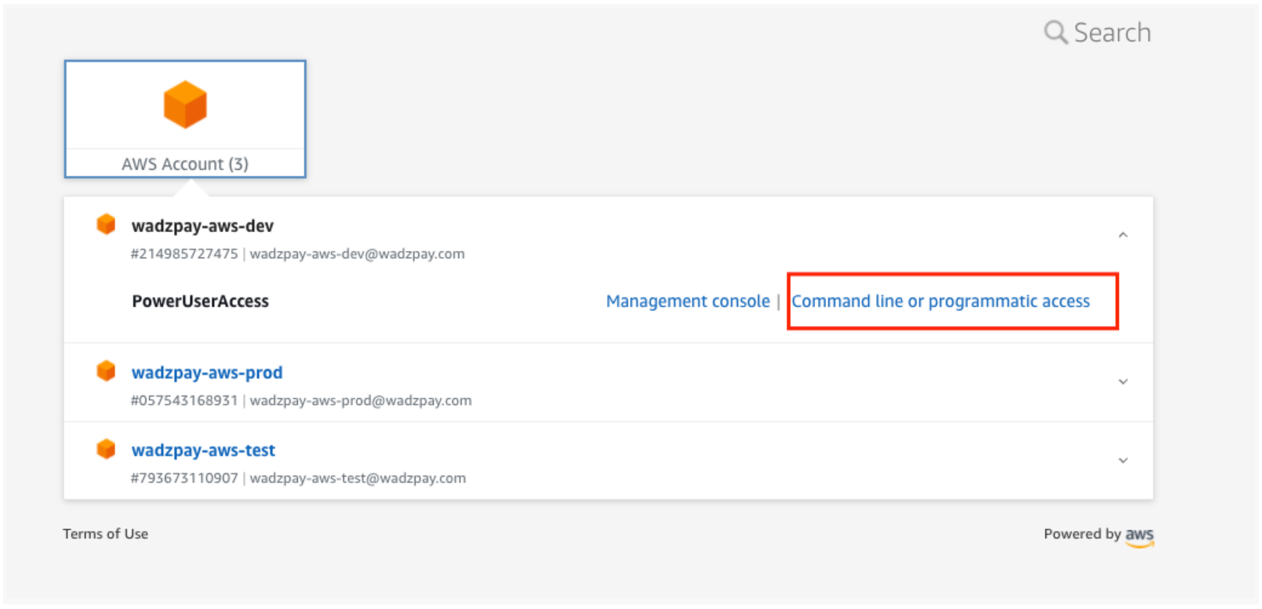
Then you can test what will be created with

```
terraform plan -var-file environments/{env}.tfvars -var-file
environments/{env}.secrets.tfvars
```

And if it looks good, you can run `terraform apply` finally with:

```
terraform apply --auto-approve -var-file environments/{env}.tfvars -var-file
environments/{env}.secrets.tfvars
```

And yeah, the most important thing, you need to put AWS credentials to your env variables:



Search

AWS Account (3)

wadzpay-aws-dev
#214985727475 | wadzpay-aws-dev@wadzpay.com

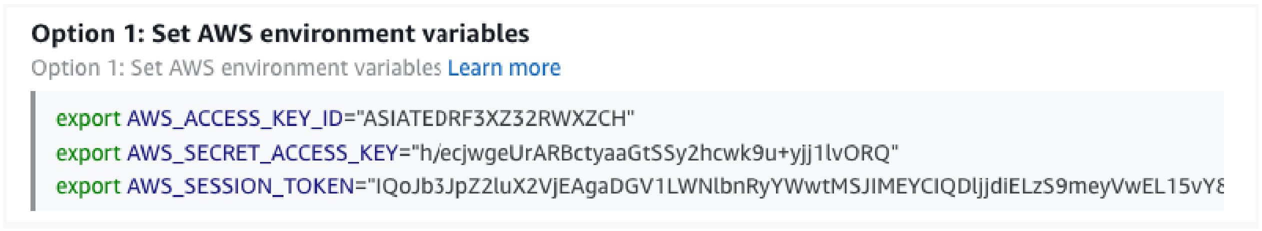
PowerUserAccess Management console | Command line or programmatic access

wadzpay-aws-prod
#057543168931 | wadzpay-aws-prod@wadzpay.com

wadzpay-aws-test
#793673110907 | wadzpay-aws-test@wadzpay.com

Terms of Use

Powered by aws



Option 1: Set AWS environment variables

Option 1: Set AWS environment variables Learn more

```
export AWS_ACCESS_KEY_ID="ASIATEDRF3XZ32RWXZCH"
export AWS_SECRET_ACCESS_KEY="h/ecjwgeUrARBctyaaGtSSy2hcwk9u+yjj1lvORQ"
export AWS_SESSION_TOKEN="IQoJb3JpZ2luX2VjEAgaDGV1LWNlbnRyYWwtMSJIMEYCIQDljjdiELzS9meyVwEL15vY
```

Click (copy) and paste to your bash

It has timeframe validity so it might happen you will be logged out after some time.

4. How to add the new variable

1. Create a new parameter in `variables.tf` root directory
2. If the variable is sensitive and needs to be stored in Parameter Store and shipped to the container as a secret environment variable, add it also to `secrets.tf` in the root directory
3. Pass that variable to fargate module, so edit `main.tf` in the root directory and `fargate/variables.tf` in the fargate module, then just edit `fargate/main.tf` and put the variable to `environment variables` or `secrets` (if you wish to encrypt it)

5. How to deploy the last update to production

First, you have to specify the correct variables

Among those we don’t have appconfig_wallet_ids_btc, appconfig_wallet_ids_eth, appconfig_wallet_ids_usdt. These are production BitGo wallet ids and you should ask Simon for them. App BitGo token which allows sending app_bitgo_token. Appconfig_wallet_passphrases_btc, appconfig_wallet_passphrases_eth, appconfig_wallet_passphrases_usdt are passwords for production wallets and Simon should know them. Once you have all these variables, you can proceed to change them as described in pt. 3. Then you can deploy the newest code as described in pt. 2. Then you have to authorize in Swagger and call **/migrateSubaccounts**, **/migrateAddresses**, **/createWadzpayMerchantAccounts**. Once you are done, it is better to remove these endpoints, so no one could call them.

Possible risks:

1. USDT was not tested on testnet
2. Address creation goes wrong as there are a lot of accounts to create addresses on production so we should see how BitGo will react to a much higher load there.
3. BitGo ETH wallet is dead for the last three days on testnet, so there can be some troubles on production as well.
4. Fee estimation endpoints by BitGo don’t estimate fees well. To be precise, there is a big risk if the fee is underestimated two times than the actual fee.

.