



Unicorn.Rentals

Hippogriff Stables (?) Operational Runbook

Document Control

Revision	Author	Date	Description
1	Mike	3/1/2015	draft!
2	Dave	3/14/2015	draft: major updates and fixes
1.0	Dave	3/21/2015	final draft! So excited!
1.1	Mike	3/24/2015	major fixes. Added Troubleshooting after fun with disk drives :-)
2.0	Dave	4/1/2015	migrate to AWS. Removed Troubleshooting: no more physical hardware!
3.0	Mike	4/2/2015	full rewrite of 2.0. Added Troubleshooting again. Dave is an optimist.
4.0	Dave	5/13/2015	full rewrite of 3.0. Added Disaster Recovery for obvious reasons, Mike.
4.1	Mike	5/13/2015	minor updates to DR section based on the Event as mandated by mgmt.
5.0	Dave	5/16/2015	4.1 was a tissue of lies. Rewrote Disaster Recovery based on the latest Event. Last commit before the long nap.

Introduction

The Unicorn Rentals website provides cryptographically strong hashes as a service. The system is comprised of a client application and a standalone server. The system will always return the same hash value for a given key. The system builds the hash based on a proprietary, computationally expensive hashing algorithm.

The server is currently deployed as a single, standalone 'go' binary. There is no session state to track or external calls to make so the server architecture should respond well to simple horizontal scaling. As there is expected to be a constrained set of client keys, the system should respond well to caching techniques.

That's it for hints.

Scope

This runbook describes the operational theory and practice for the production system powering the Unicorn Rentals website. The primary audience is the DevOps team running the site. The DevOps team is responsible for deploying code, scaling the site in response to load, maintaining our published SLA's (including response time and uptime), disaster recovery, troubleshooting activities and any monitoring and alerting activities required to meet these objectives.

Infrastructure Cost

Here at Unicorn Rentals we know that it takes money to make money. You will be charged \$8 for every 15 minutes of EC2 usage. For example, if you have one EC2 instance running for 3 minutes and terminate it, you will be charged \$8. If you have 2 instances running for 13 minutes, you will be charged \$16. So make sure you only provision what you need!

Player Dashboard

The dashboard can be accessed by going to <https://dashboard.eventengine.run>. It will prompt you to enter your Team Hash. This team hash can be found on a piece of paper handed to you earlier today.

Once you login to the dashboard, it's time to set your team name. Please click on the "Set Team Name" button on the dashboard and enter a team name. Keep in mind that this team name cannot be changed once set and will represent your team on the scoreboard. The scoreboard team at Unicorn.Rentals did not have the motivation to make the scoreboard private. It is a public site and team names should be created with that in mind. Any inappropriate team names will be renamed by the Unicorn.Rentals board of directors.

Dashboard

Dashboard Guide

Help

Logout

SCORETREND

RANK

633.67103.671

Game

Score Events

Scoreboard

Chat Invite

AWS Console

Game: gd2015-doc-test

Team Name: test

Game ID: 41745e7f99a124af9b621566c7c59b00

Team ID: c53df11fb7af465e997906ab5dfdc6e

Modules

Order Processor

Score: 1,027 / Trend: 169

Outputs:

No outputs defined

Inputs:

Server Address

Current Value: http://52.72.237.43

Update

Mini Games

Can Kyle Sandiego order unicorns by text from anywhere in the world?

todo

Instructions

Answer

Code

The dashboard has a few key components that you will interact with throughout the game:

- The top bar of the dashboard has a series of buttons that allow you to: 1) Access your score events, 2) Access the scoreboard, 3) Set your team name (can only be done once), 4) Request invite to Slack channel for team collaboration and chatting, and 5) Access your team's AWS account (--- this is important!)
- Below this bar you will see a list of modules enabled for this game. Each module can define outputs and inputs. In the example above, the module has no outputs, but does have one input. For this module, you will provide a URL to your order processor so we can send you order (and you can get points).

NOTE: Once you set your team name, and your team name is approved by the UnicornRentals collective, additional options will appear on the dashboard. These options include your team score, rank, and trend, as well as access to the scoreboard via the "Score Events" button.

Score Events and Scoreboard

If you click on the "Scoreboard" link on the Player Dashboard, it will open a window to the Scoreboard. The Scoreboard is where you will be able to track your progress as a team against all the players in the room.

To get a deeper view on the performance of your individual team, you can click on the "Score Events" button on the player dashboard to access your point-by-point breakdown.

Points	Total	Source	Reason
-1	531.22	Order Processor	Request error: Get http://52.72.237.43/calc?input=5UNhbkhhelVuaWNvcm%N2FLTowMc%3D: dial tcp 52.72.237.43:80: i/o timeout
-1	532.22	Order Processor	Request error: Get http://52.72.237.43/calc?input=5UNhbkhhelVuaWNvcm%N2FLTETNTR%3D: dial tcp 52.72.237.43:80: i/o timeout
-1	533.22	Order Processor	Request error: Get http://52.72.237.43/calc?input=5UNhbkhhelVuaWNvcm%N2FLTISODc%3D: dial tcp 52.72.237.43:80: i/o timeout
-1	534.22	Order Processor	Request error: Get http://52.72.237.43/calc?input=5UNhbkhhelVuaWNvcm%N2FLTvhNQK%3D: dial tcp 52.72.237.43:80: i/o timeout
-1	535.22	Order Processor	Request error: Get http://52.72.237.43/calc?input=5UNhbkhhelVuaWNvcm%N2FLTpnN%3D: dial tcp 52.72.237.43:80: i/o timeout
-1	536.22	Order Processor	Request error: Get http://52.72.237.43/calc?input=5UNhbkhhelVuaWNvcm%N2FLTlyM%3D: dial tcp 52.72.237.43:80: i/o timeout
-1	537.22	Order Processor	Request error: Get http://52.72.237.43/calc?input=5UNhbkhhelVuaWNvcm%N2FLTMDM%3D: dial tcp 52.72.237.43:80: i/o timeout
-1	538.22	Order Processor	Request error: Get http://52.72.237.43/calc?input=5UNhbkhhelVuaWNvcm%N2FLTQMD%3D: dial tcp 52.72.237.43:80: i/o timeout
-1	539.22	Order	Request error: Get http://52.72.237.43/calc?input=5UNhbkhhelVuaWNvcm%N2FLTpnNZY%3D: dial tcp 52.72.237.43:80: i/o timeout

This page has two sections to pay attention to:

- Each row lists every score event that your team has generated. The "Source" column tells you where the point awards or deductions came from. The "Points" column will tell you how many points you got or lost.
- The "Reason" column will tell you the reason you got the points or lost the points. Pay very close attention to this column when you are losing points in order to understand what is going on and how to fix the problems.

References

- The server application is deployed,as a 'go' binary compiled from source rumored to be stored in a github repository. However, the name of this repository is unknown to the current operations and development staff.
- The server application can handle about 5 connections before starting to get really slow. Be careful about overloading, and watch for 503s when the queue fills. You can try restarting the app or server if it starts to get backed up.
- The server application is an x86 statically linked, unstripped ELF executable found here: <https://s3.amazonaws.com/ee-assets-prod-us-east-1/modules/gd2015-loadgen/v0.1/server>

- The base OS we have chosen is Amazon Linux. This distribution was selected for it's broad industry support, stability, availability of support and excellent integration with AWS. This distributions was selected by SecOps based on their requirements for platform hardening.
- Architecture was moved to AWS as part of go-to-market plan. Operating the AWS CLI (<http://docs.aws.amazon.com/cli/latest/Userguide/installing.html>) might be helpful.

- When working with AWS, only the following roles are allowed by SecOps... and finance:
 - ec2
 - s3
 - elbs
 - ecs
 - eks
 - cfm
 - elasticache
 - cloudwatch
 - sns
 - systems manager
 - cloudtrail
 - config
 - vpc
 - cloudfront
 - lambda
- Getting bored of writing this silly thing. Who needs it?

Application System Architecture

Client <-> Server: Pretty sure anyway. What else is needed here?

Unicorn_Data_Flow

On-line Operations

Update Application

- Our Server code is downloaded on first-boot via a script provided by AWS as 'User Data'
- The User Data is stored in the Auto Scaling Group 'Launch Configuration':

- \$ aws autoscaling describe-launch-configurations --query LaunchConfigurations[0].UserData | base64 --decode

```
#!/bin/bash
wget 'binary location (reference 3)'
chmod +x server
./server
# Reboot if the server crashes
shutdown -h now
```

Here is an overview of the AutoScaling Group Launch Configuration:

- Make a new Launch Configuration based on the old one (<http://docs.aws.amazon.com/AutoScaling/latest/DeveloperGuide/WorkingWithLaunchConfig.html>)
- Update the location of the server code downloaded from S3 in 'User Data' section.
- Associate this new launch configuration with the Auto Scaling group.
- Scale-up new group / scale-in old group
- Make sure the instance can see the S3 bucket on launch (default route via IGW in subnet).

Add ssh-key to instance

- Create an ssh-key-pair (<http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-key-pairs.html>)
- Update Launch Configuration (Same procedure as 'Update Application')
- Yes, this means you must relaunch the instances.

Update Auto Scaling Policy

Never tried this! Check out http://docs.aws.amazon.com/AutoScaling/latest/DeveloperGuide/scaling_typesof.html

Troubleshooting Procedures

Networking walkthrough

The AWS VPC / ELB environment must be healthy for the application to work. All production traffic flows through the ELB on both ingress and egress:

Unicorn_Detail_Network

- Check the Security Group settings for your instances
- Make sure all required ports are Allowed
- Check the Routing tables on your subnets
 - Make sure the routing tables are applied to each subnet
 - 'Default' table applies to all subnets without an explicit definition
 - Make sure the routing table has the appropriate rules
- Things to check in the the VPC that Ops has broken in the last six - 12 months.
 - Instances up? Try 'http://<<Your-IP-or-DNS>>/healthcheck' to for more detail
 - Instance 'up' in the Auto Scaling group?
 - Subnets?
 - Subnet 'cider'??? What did that guy say it was called? Is it big enough???
 - Are the subnets added to the Elastic Load Balancer?
 - Are the subnets added to the Auto Scaling Group?
 - Routes correct / intact? See diagram.
 - ACLs: set on subnet. Too restrictive/permissive?
 - Security groups?
 - IGW? Do we have routes to flow traffic through the IGW? Required to grab the server code from S3.
 - Route53 records? Are the records pointing to the correct resources?

- New application test utility!

<https://ee-assets-prod-us-east-1.s3.amazonaws.com/modules/gd2015-loadgen/v0.1/server-bang-linux>
<https://ee-assets-prod-us-east-1.s3.amazonaws.com/modules/gd2015-loadgen/v0.1/server-bang-osx>
<https://ee-assets-prod-us-east-1.s3.amazonaws.com/modules/gd2015-loadgen/v0.1/server-bang-windows>

```
./server-bang.$arch --server URL
```

- You can try ssh'ing to the node and checking out the application --

It runs on port 80: thats about all I know. You must install an ssh key first (see 'Add ssh-key to instance', above)

- As mentioned, the server process can get slow if it is handling too

many connections. Try restarting.

Change Management

HAHI!

System Monitoring

- Application monitoring: <https://scoreboard.eventengine.run>
- How to check ELB metrics?
http://docs.aws.amazon.com/AutoScaling/latest/DeveloperGuide/policy_creating.html
<http://docs.aws.amazon.com/ElasticLoadBalancing/latest/DeveloperGuide/elb-cloudwatch-metrics.html>

To Do

- Explore elasticache, cloudfront: Didn't Jayne say this might speed things up?
- How to test HA? Automate DR.
- Refactor infrastructure using container architecture