TrackIt

aws & Magento

Application Implementation Note

Magento®, an Adobe company, is a Cloud eCommerce Platform that TrackIt has successfully implemented utilizing a variety of Amazon Web Services. Technical challenges were discovered and overcome during the project. Technical details are outlined in this application note.

TABLE OF CONTENTS

Services Being Hosted

- The client website, a Magento ecommerce store, written in PHP
- The Product Information Management (PIM) system, which provides manipulation of the product sets without requiring access to the store backend
- Client's Custom Connection application, which is a bridge between the store, the marketplace and the logistics environment (warehouse, shipping orders, etc.)
- The Mirakl-HiPay connector that integrates the cash-out operations between HiPay and Mirakl
- Client's Custom application that processes orders and prints shipping labels

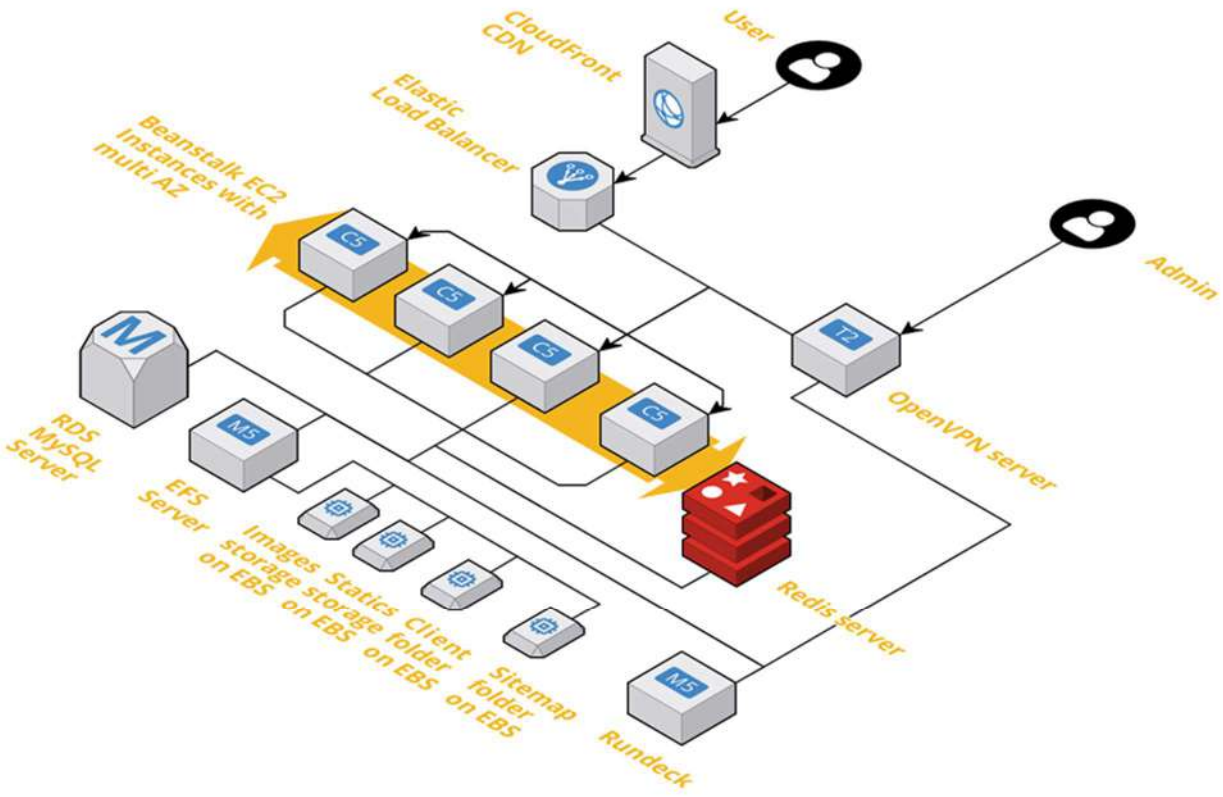
Services Utilized

- Data storage for all media and statics are hosted on an NFS server distributed to the Magento instances
- The front end and back end of Magento is hosted on AWS Elastic Beanstalk, with separate development and a production environments
- The PIM, Custom Connection application, and Mirakl connectors are hosted on Beanstalk
- The Redis in-memory data store is hosted on AWS ElastiCache
- AWS CloudFront is utilized as the Content Distribution Network (CDN)
- Databases are hosted on AWS Relational Database Service (RDS)
- OpenVPN, Ansible, Rundeck, Microsoft Windows Servers, Gateway Servers and phpMyAdmin are hosted on AWS EC2

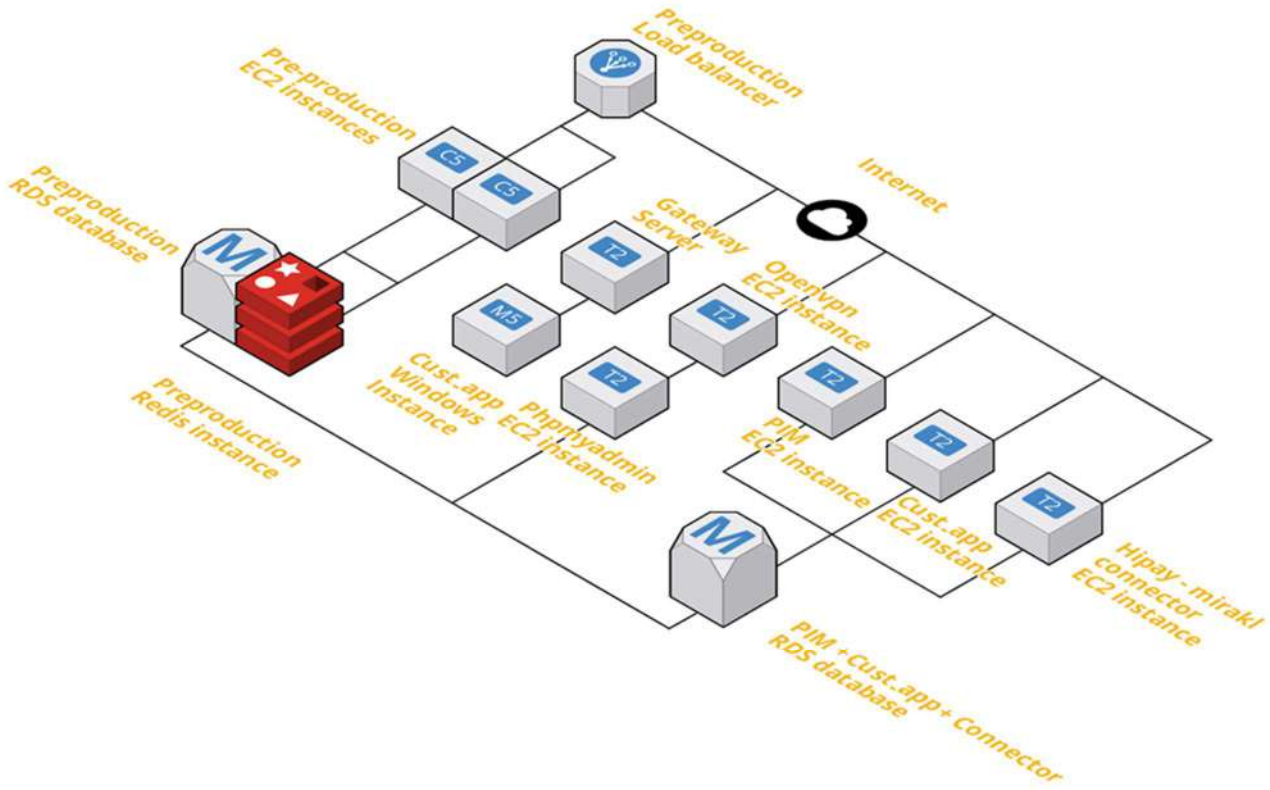
EC2 (and operating environments) is a client-managed service; all other services are managed by AWS

Infrastructure Diagrams

Magento Production Environment diagram



Supporting Environment Diagram



Services Detail

1. Magento

The Magento software is deployed and managed by Beanstalk. Two nearly identical environments were created, one for pre-production and one for production.

In the production environment there are two beanstalk environments. The first is for the front end servers that serve traffic to clients. The second is for the back office server which handles tasks such as re-indexing, product import, routines, and the access interface to the backend web.

There are four front end servers and one backend server. They are in two different target groups, and are both served by a single Application Load Balancer. This ALB uses a path-based routing rule, which directs all traffic on /backend/* to the back office server, and the rest to the frontend server.

The code is kept in sync between the front end and back office server.

The source of the code is in a repository handled by the client.

Beanstalk configuration files, that lists all the actions performed by the instance during deployment, are stored in this repository "git@bitbucket.org: client/beanstalk-config.git", on the branch "prod" for production, and "preprod" for pre-production.

The configuration file with the secret parameters (keys, database connections, etc) is kept on AWS S3, and was pulled by the instances during the deployment.

The Magento instances communicates with these services:

- RDS: connects to the production database endpoint on RDS
- NFS: supplies 4 different mount shares:
  - /var/app/current/pub/media: Media files for the website
  - /var/app/current/pub/static: Statics files for the website
  - /var/app/current/sitemap: Sitemaps of the website
  - /var/app/current/client: Orders that are exported out of the website for handling
- ElastiCache: Connects to a Redis cluster to handle website caching for performance

The deployment process was handled by Rundeck. There is not automatic deployment when a new commit is pushed, but it is a "one button" deployment.

Rundeck deployment process:

- Latest version sources are cloned (both Magento code, and Beanstalk configuration files)
- .user.ini and .htaccess files are modified
  - Adjust RAM limitations
  - Change some redirection paths
- Merges the configuration files with the code and delete the .git directory
- Deploys to Beanstalk

Beanstalk deployment process:

- Beanstalk receives the new version of the code and configuration as a zip file
- Configured to deploy in a rolling methodology, one server at a time
  - Each server is individually removed from the load balancer, and waits for connections to drain
  - The deployment commands are run by Beanstalk start, removes any old code, and replaces with the new version
  - Magento commands are run to generate the statics, disable/enable modules, and setup new database modules
  - Once the deployment is complete on the instance, it is reattached to the load balancer and waits for health checks to be passed from both Beanstalk and the load balancer
  - If the instance passes as healthy, deployment commences on the next instance
- In the case of an error during a deployment (any unsuccessful command), it is automatically aborted by Beanstalk, and the affected instance are not reattached to the load balancer
  - These instances are left as-is for debugging purposes

2. PIM

The PIM is deployed on Beanstalk, but the deployment is much simpler. The deployment only requires:

- Setting up the credentials
- Updating the DB scheme from the code
- Chowning the repository to the correct user permissions

The PIM Beanstalk configuration can be found on the same repository as Magento's, except it is in the branch master in the "pim" directory.

There is only one instance being deployed, as high availability is not required.

The instance is accessible from the internet via the load balancer DNS name, in HTTP only.

It is connected to an RDS instance database, shared with the Custom Connection application and Mirakl connector.

3. Custom Connection Application

The Client's in-house developed Custom Connection application deployment is almost identical to the PIM, except it also has an ftp server, used to transfer product images and other items to the server.

The instance is not directly publicly accessible as it goes through a gateway server (detailed later in this document) that only exposes the required ftp ports.

The ftp user and password are created during the deployment, as well as the vsftpd configuration.

The Beanstalk configuration files can be found in the same repository and branch as the PIM, but in the Custom Connection application' own directory.

The web root directory (/var/app/current) needs to be accessible, and writeable, from ftp. Since the web user is distinct from the ftp user, the directories in the web root directory need to be executable by anyone in the web user group, so they are chmoded to 775.

The instance is accessible from the internet via the load balancer DNS name, in HTTP only.

It is connected to an RDS instance database and shared with the PIM and Mirakl connector.

4. Mirakl-HiPay Connector

The Mirakl-HiPay connector deployment is almost identical to the PIM and Custom Connection application deployments.
The configuration file must be added, along with the Apache configuration. A cron operation is also added.

The Beanstalk configuration can be found in the same repository and branch as the Custom Connection application, except in the "mirakl-cashout" directory.

The instance is accessible from the internet via the load balancer DNS name, in HTTP only.
It is connected to an RDS instance database, shared with the Custom Connection application and PIM.

5. Custom Application (client developed)

The Custom application, developed by the client, is neither managed nor deployed via an AWS service. It runs on a Windows instance and all the software in it was installed manually.

There are two attached EBS volumes. Some of its ports are accessible from the internet via the gateway server. Those accessible ports are for RDP from the internet, and access to an ftp server running on the instance.

6. Non-AWS Managed Hosted Services

The following services are managed by Packer and Terraform. Packer builds the AMI to run on those servers, and Terraform launches those servers.

a. OpenVPN

The packer configuration will first copy private keys to the instance (private keys are persistent such that if the image needs to be updated the same keys are used), then an installation script is run to generate client configurations.

The client creation script is located in /root/client-configs/create_user.sh. The only parameter required by this script is a unique username.

An Elastic IP is attached to the instance.

b. Rundeck

There are two separate parts to the Packer configuration for Rundeck.

Rundeck must have persistent data for its different jobs and configuration, and the data cannot be lost if the instance is re-deployed. The Packer AMI does not allow nor provide persistent data on its instances however. Therefore, all of the persistent data for Rundeck is stored and maintained on a mounted AWS EBS volume.

First, there is a Packer configuration to install Rundeck itself on an instance, and to then build an AMI for it. There is nothing particularly custom added here, only the specific dependencies for Rundeck.

Second, there is a Packer configuration to deploy the EBS volume attached to the instance that stores the persistent data for Rundeck. This configuration is only needed for bootstrapping the EBS volume, as once it is configured it should not have to be restarted from scratch.

The web portal for Rundeck is accessible on the internal domain name (rundeck.internal-client.url) on port 4440. It is only accessible from inside the VPC, with no external access allowed.

c. EFS Server

The Packer configuration for the EFS server is relatively light, as it only needs to install its specific dependencies.

The actual configuration of shares, and mounting of EBS volumes, is performed by a user_data script in the Terraform configuration. This script attaches the EBS volumes to mount points, and shares those mount points in the exports file.

The exported shares are the following:

- client-media: share that contains all of the media for the website
- magento-statics: share that contains the generated statics, that must be accessible by all instances
- magento-client: share that contains the order exported from the website
- magento-sitemap: share that contains the sitemap of the website

d. Gateway Server

The gateway server is an instance running HAProxy. It provides public access for certain services, without completely exposing the instance.

The exposed services are all running on a Windows instance:

- SMB shares: port 445 is proxied to mount SMB shares from outside the network
- Custom application: ports 8700-8800 are proxied to allow external communication
- RDP access: port 3389 is exposed to connect to the server from outside via RDP
- FTP access: ports 21, 2048-4096 are exposed to connect from outside to the FTP server. Port 21 is for the actual connection, and the range of ports are for the PASV mode.

The packer configuration is fairly trivial. Tt installs the HAProxy packet, and copies the configuration file.

For security purposes, the instance is not completely open to the internet, the security group is a whitelisting of IP addresses.
The IP addresses authorized to access this instance are the following :

- XXX.XXX.XXX.XXX
- Etc.

7. AWS Services

a. RDS

Three different databases are utilized. All are accessible only from inside the VPC.

The largest (in this case two r4.4xlarge instances) is the production database. These instances implement high availability by utilizing synchronous replication to another instance in another availability zone. Only the production website connects with the production database.

The second is the pre-production database. It is significantly smaller than the production database with only one r4.large). Only the pre-production website connects to it.

The third database shared is by the PIM and the Custom Connection application.

For each database, a snapshot is performed daily and retained until the next snapshot.

b. ElastiCache

Elasticache is implemented for two Redis clusters; one cluster for the production environment and one for the pre-production.

The production Redis cluster is designed for high availability, with a primary node and a read replica node. Automatic failover will occur if the primary fails.

The pre-production cluster does not have a high availability implementation.

c. CloudFront

Cloudfront is used as a CDN for the media and static files of the website. Two distributions are configured: one for the media files and one for the static files.

Both distributions are configured according to Magento recommendations for Cloudfront.

They both use only HTTPS, and have domain names in the form .client.url.

Both distributions point to the production instances load balancer, but with /pub/static or /pub/media added to the url.

d. S3

S3 is used to store multiple type of data. None exposed publicly.

- client-billing: contains billing document for analysis by Trackit
- client-magento-config: contains the secrets configuration files for Magento
- client-magento-elb-logs: contains the production elb logs
- client-temp-storage: used if needed to backup an archive temporarily when doing an operation. Anything in this bucket is considered temporary storage, nothing that is business-critical should be stored in this bucket
- client-terraform-remote-state-storage: stores the remote state for Terraform
- Elasticbeanstalk-XX-XXXX-X-XXXXXXXXXXXX: used and managed by Beanstalk to store the deployment's zips files

e. Certificate Manager

A SSL certificate for the domain *.client.url is imported to the Certificate Manager. This certificate is added to the CloudFront distribution, and to the production load balancer.
Since the domain name .client.url is not managed by AWS, this certificate was simply imported, not generated.

f. CloudWatch

Cloudwatch is used for monitoring of the production environment. The monitoring is split in two parts; alarms, and a monitoring dashboard.

All metrics that AWS generates on from services, as well as custom ones, are accessible from CloudWatch.

When alarms are triggered, an email is sent to a mailing list so that actions can be taken to resolve the problem.

Those alarms are as follow:

- DB CPU Prod: will send an alert if the CPU usage of the production database exceeds 70% for 3 datapoints within 3 minutes
- awsrds-client-magento-prod-High-DB-Connections: will send an alert if the number of concurrent connections to the production database exceeds 100 for 3 datapoints within 3 minutes
- Prod eb health: will send an alert if the Elastic Beanstalk health for the production environment goes above 0 for 3 datapoints within 15 minutes. The Beanstalk health is a metric generated by Beanstalk that uses different metrics gathered from the environment to determine its health. Those metrics include:
    - The CPU utilization of the instances (if they are close to capacity)
    - The type of requests coming out of the instances. If for example an instance starts to return a lot of 5xx requests, the health of the environment will degrade
    - The latency of the requests. If an instance doesn't respond, or starts taking a long time, the health of the environment will degrade
- DLM-[Lan/Ml/Cetsi] VPN status: monitors the three IPSec VPN tunnels, and will send an alert if one of the tunnels is down for more than 15 minutes

The monitoring dashboard provides representation of the important metrics of the production environment at a glance.

Metrics displayed in the production dashboard:

- Environment Health: displays the current Elastic Beanstalk health
- Application requests stats: aggregation of multiple metrics coming from the production Elastic Load Balancer, on a one-hour time period
    - Request count: sum of all the requests
    - HTTPCode_Backend_5XX: sum of the all the 5xx requests returned
    - HealthyHostCount: average of number of hosts (instances) considered healthy by the Load Balancer. A healthy host is a host that passed the Load Balancer healthcheck, which is currently an HTTP GET on the page /pub/health_check.php, that will return a 500 if Magento is missing something to make it work, such as the mysql connection, or the Redis connection.
    - UnHealthyHostCount: same as the above, but it displays the number of unhealthy hosts
    - Latency: average of the latency to complete a request
    - EstimatedALBActiveConnections: average of the estimated number of TCP connections to the Load Balancer
    - EstimatedProcessedBytes: sum of the estimated processed bytes by the Load Balancer
- LoadAverage5min EC2 instances: graph of the 5-minute average CPU load of the four production instances
- RDS instance CPU utilization: graph of the average CPU utilization as a percentage of the production database instance
- Backend latency: graph of the average latency to complete a request
- CPU usage Redis: graph of the average CPU utilization as a percentage of the Redis instance
- Backend request status: stacked area graph of the request types (3xx, 4xx, 5xx) over the previous hour

g. Networking Setup

All services and instances are in the same VPC, which has the following IP range: XXX.0.0.0/16

There are 6 subnets in the VPC, 2 per availability zone; one public and one private.

The public subnets are:

- XXX.0.1.0/24
- XXX.0.2.0/24
- XXX.0.3.0/24

The private subnets are:

- XXX.0.101.0/24
- XXX.0.102.0/24
- XXX.0.103.0/24

The VPC has an Internet Gateway to communicate with the external Internet, and each private subnet has a NAT to provide connectivity to the internet.

By convention, all instances that do not require high availability are hosted on the first subnet (XXX.0.1.0/24, XXX.0.101.0/24); either the public or the private one.

There are also three VPN IPSec links that are connected to different parts of the client infrastructure and are needed to access different services.

There is also a VPN access to this network via an OpenVPN server. All subnets are accessible when connected to this VPN.

h. Domain Names

AWS Route 53 is used to manage the DNS of two domain names. The first one is .internal-client.url. which is used for internal domain names (e.g. rundeck, windows instance, nfs server), and is not publicly advertised (it is only advertised by the internal AWS DNS servers). This domain is registered with AWS.

The second one is the .client.url. domain. The ownership of the domain is not AWS.

Subdomains in use for the domain .internal-client.url.:

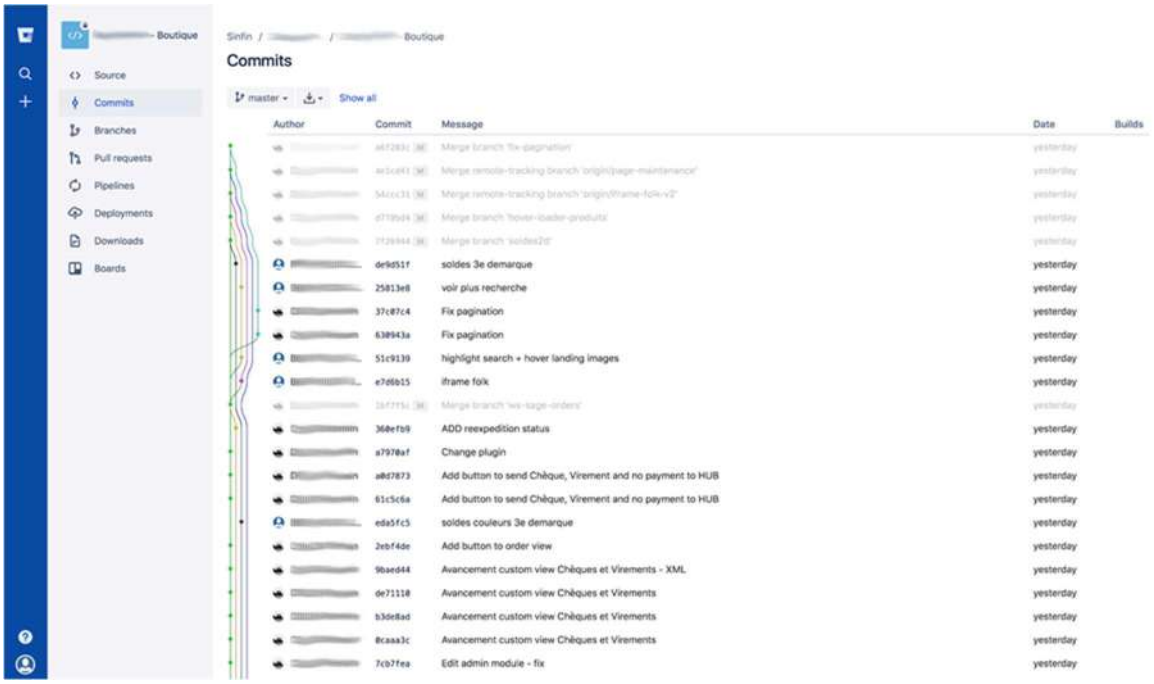
- nfs.internal- client.url
- rundeck.internal- client.url
- windows.internal- client.url

Subdomain in use solely for the AWS infrastructure, for the domain .client.url.:

- client.url
- www.client.url
- media.client.url
- Static.client.url

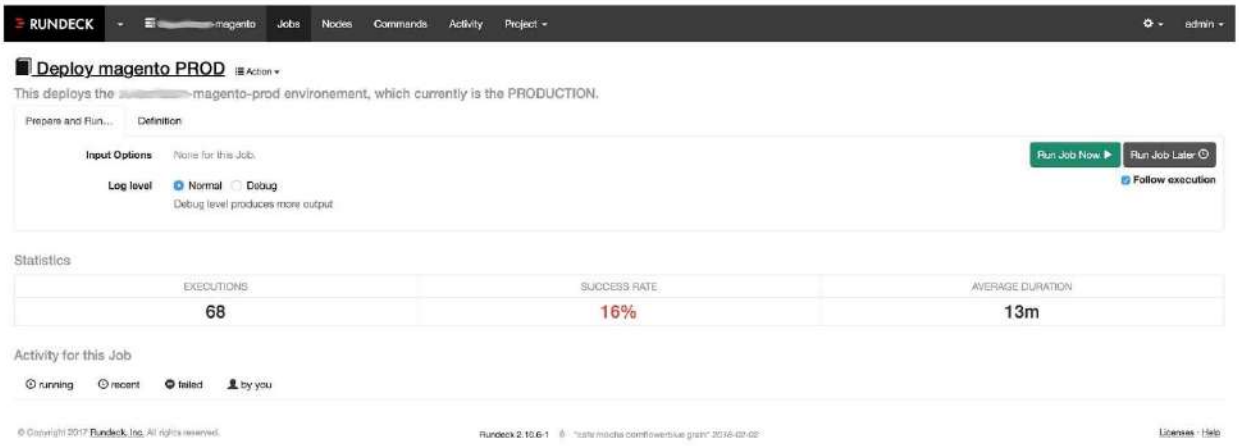
Deployment Process for the Production Environment

- The master branch of the Magento repository is checked to ensure it has the correct version to deploy. The same is done for the repository of the Beanstalk configuration on the master branch.

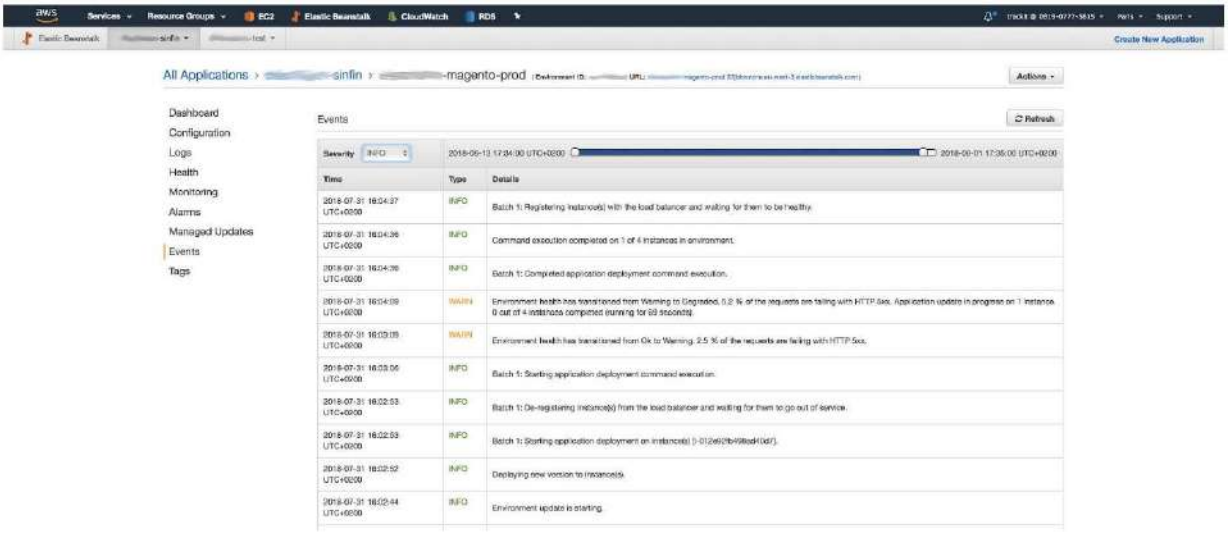


- The store is put into maintenance mode. The following command is run on the four instances : "*sudo -u webapp php /var/app/current/bin/magento maintenance:enable*"

- Connect to Rundeck at the following address: http://rundeck.internal-client.url:4440/menu/home, go to the project "*client-magento"*, then in the job "*Deploy magento PROD"*



- Launch the Rundeck job. Once the deployment has started, an overview of the deployment progress can be monitored on the Beanstalk console.



- Once the deployment is over, the statics needs to be generated. To do so, go to any production instance and delete the content of the folder */var/app/current/pub/static*, except for the *.htaccess* file. The command to do so is *sudo rm -rf /var/app/current/pub/static/*.* Then run these commands to generate the statics :
    - *sudo -u webapp php /var/app/current/bin/magento setup:static-content:deploy --force --area frontend --theme Client/dlm fr_FR*
    - *sudo -u webapp php /var/app/current/bin/magento setup:static-content:deploy --force --area adminhtml en_US fr_FR*
- Flush the cache. To do so, run this command on the four production instances: "*sudo -u webapp php /var/app/current/bin/magento cache:flush*"
- Disable the maintenance mode on the store. To do so, run this command on the four production instances: "*sudo -u webapp php /var/app/current/bin/magento maintenance:disable"*

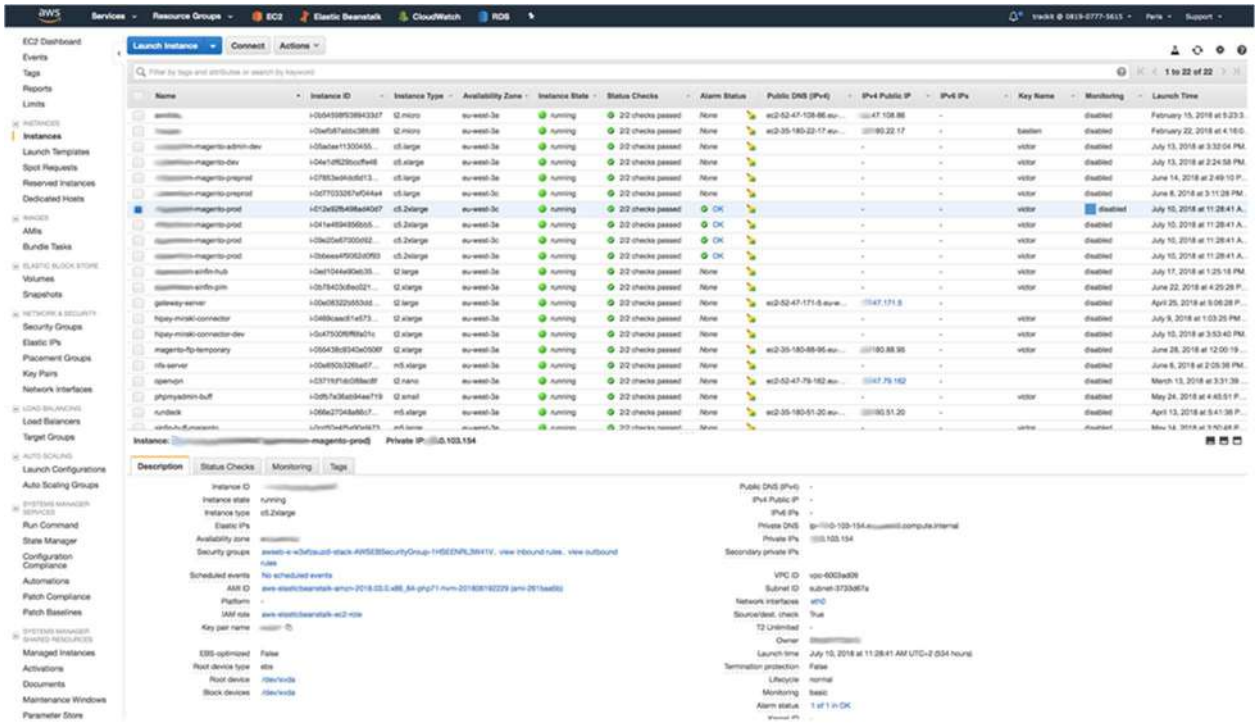
In Case of an Error

If an error happens during a deployment, Beanstalk will automatically stop the deployment and will not re-attach the instance to the Load Balancer. Beanstalk will print details about the error (such as the instance id on which the error happened), and a truncated version of the error message in the log file.

2018-07-19 15:44:17 UTC+0200	ERROR	During an aborted deployment, some instances may have deployed the new application version. To ensure all instances are running the same version, re-deploy the appropriate application version.
2018-07-19 15:44:17 UTC+0200	ERROR	Failed to deploy application.
2018-07-19 15:44:16 UTC+0200	ERROR	Unsuccessful command execution on instance id(s) 'i-012e92fb498ad40d7'. Aborting the operation.
2018-07-19 15:44:16 UTC+0200	INFO	Command execution completed on 1 of 4 instances in environment.
2018-07-19 15:44:16 UTC+0200	INFO	Batch 1: Completed application deployment command execution.
2018-07-19 15:44:16 UTC+0200	ERROR	[Instance: i-012e92fb498ad40d7] Command failed on instance. Return code: 1 Output: (TRUNCATED)...o_Customer': Module 'Magento_AdminNotification': Module 'Magento_Indexer': Running data recurring...Warning: PDOStatement::execute(): MySQL server has gone away in /var/app/ondeck/vendor/magento/zendframework1/library/Zend/Db/Statement/Pdo.php on line 228. container_command 020_install_magento in .ebextensions/01_setup.config failed. For more detail, check /var/log/eb-activity.log using console or EB CLI.
2018-07-19 15:39:48 UTC+0200	INFO	Batch 1: Starting application deployment command execution.
2018-07-19 15:39:47 UTC+0200	INFO	Batch 1: Starting application deployment on instance(s) [i-012e92fb498ad40d7].
2018-07-19 15:39:46 UTC+0200	INFO	Deploying new version to instance(s).
2018-07-19 15:39:19 UTC+0200	INFO	Environment update is starting.

To troubleshoot the problem, reconnect via SSH to the instance on which the error happened. To find the IP address of the instance, get the instance id given by Beanstalk (for instance i-012e92fb498ad40d7). Then go into the EC2 console:

e.g.: https://eu-west-3.console.aws.amazon.com/ec2/v2/home?region=eu-west-3#Instances:sort=tag:Name, select the instance with the corresponding ID, and retrieve the private IP of the instance.



Tail the log file (via the command tail) /var/log/eb-activity.log. This file logs all the commands executed by Beanstalk during a deployment.

If there was an error during the deployment, it will be logged in this file.

After having found and corrected the cause of the error, relaunch the deployment in the same manner as before, via Rundeck.

Issues and Shortcomings

Specifically Beanstalk related to high availability:

Blue/Green Deployments Not Practical

When deploying a new Magento website, databases migrations are frequent. Blue/Green deployment are thus very difficult, since a migration from the new environment will impact the old deployment, and break the website.

It would be possible to have one database per Blue/Green environment, but this adds significant complexity due to the requirement to sync databases that have diverged; especially difficult when handling payments, so it's not really feasible.

Therefore deployments require downtime.

Since deployments are performed sequentially on each host, the website needs to be in maintenance mode when occurring, again, since online database upgrades would break the website. It should also be noted that maintenance mode is a database setting, so it cannot be activated/deactivated for single hosts.

While loading the website when in maintenance mode, it will return an HTTP code of 5xx. This unfortunately confuses the Beanstalk health check run once a deployment is completed on each host, and the deployment will fail.

To address this issue maintenance mode must be deactivated after each host has finished its deployment. The website is available for end users, but in a broken state since the code on the host may not match the database version. The statics and cache that are generated after the deployment may be affected as well.