**Rackspace**: G&V have “Fanatical Support for AWS” from Rackspace. Rackspace has a single sign on in front of that and simply transfer the AWS account to Rackspace is a straight forward step. They have got 2 products one is “Navigator” which is lower level and other one is “Aviator” which is higher level but Our SLA tier for products is “Aviator”. This is a legacy offering so they’ve disbanded the navigator and aviator, they can be on it, if we want to on them. Aviator was the highest level of service available short of taking Rackspace up on Dedicated Resource. Full details of Aviator can be found here:

* https://www.rackspace.com/sites/default/files/white-papers/fanatical\_support\_for\_aws\_-\_service\_overview\_1.pdf

Basically In short when it comes to working with Rackspace, we tend to engage Rackspace if:

* We need new infrastructure creating ( e.g. For Hong Kong, We can reuse the production infrastructure we’ve got and we’ll be needing development and a new set of staging so that request will go into a ticket and then they’ll process it and validate the ticket in Rackspace)
* We need to make significant changes to infrastructure size or quantity at time of peak say Christmas or New-year etc.(with the exception of autoscaling group instance counts)

Rackspace will automatically engage us if there are incidents relating to our infrastructure monitoring excluding application (e.g. an Autorecovery EC2 didn’t autorecover) if Magento application goes down due to error in transaction say and if EC2 instance is healthy and then Rackspace is not going to pick up this problem. If admin instance recovery isn’t working then Rackspace will automatically jump and resolve the issue. We’ll be getting notifications for the Critical Security or Updates needed (e.g. EC2 scheduled maintenance) and Things are expiring or becoming end of life (e.g. SSL certificates)

**Rackspace Tickets and SLA**: This ticketing system is done through Rackspace instead of AWS. If Rackspace needs to raise a ticket with AWS, Rackspace will tell you and they’ll raise the ticket for us and they’ll report back. There is no direct access to AWS support, this level of contact with AWS has to go via a Rackspace ticket. If AWS raises a support ticket, it is auto forwarded into Rackspace to create a new ticket. The AWS cloudwatch will trigger notifications from AWS will gets bubbled up to Rackspace ticket. Tickets can be raised by Rackspace and aimed at us similarly we can also raise tickets and aim them at Rackspace. Generally Tickets categories can be ‘Changes’, ‘Incidents’ or ‘Service Requests’ and can be assigned to right people. Changes will be assigned to category of add/modify/remove infrastructure. Incidents will be categorized for the unplanned interruptions or reduction in service quality. Service Requests category will be assigned to the things like account access, account settings, managing users, and billing, payments and request information. Tickets need to be attached to an account (when a ticket applies to all four accounts – we list all accounts in the ticket body by listing the account name and numbers) for e.g. the accounts were as Dyson Staging, Dyson EU Production etc. The Aviator SLA maps onto ticket severities:

* Urgent: Production System Outage / Significant Business Impact [60 Min Response Time] these tickets need to be followed by a phone call to escalate and qualify for the SLA response
* High: Production System Impaired / Moderate Business Impact [4 Hour Response Time]
* Normal: Issues and Requests / Minimal Business Impact [12 Hour Response Time]
* Low: General Information, Questions, and Guidance [24 Hour Response Time]

We need to be more proactive for the Incident kind of category irrespective of severity compared to change/request categories.

**Accessing AWS – Console Access**: AWS console is only accessible through the Rackspace single sign on (SS0) as they haven’t provided access directly to AWS console. User management is done through Rackspace (instead of directly to AWS console) and maps onto AWS console access too. In Rackspace User details section, we need to specify the contact type say (Administrative, Technical etc.) so depending upon the ticket severity we can assign tickets to the person based on contact type. If calling Rackspace about an existing or to raise a new ticket the per-user support pin needs to be given over the phone as authentication. The support pin can be found in support section of Rackspace.

**Accessing AWS - Common tasks (Developers)**: There is a lot of stability in the AWS account from a day to day perspective. Most development engineers only really need to look at EC2 inventory information or make changes to security groups controlling IP Allow listing. They can view the deployments and had a permission to scale-up the instances if needed. Otherwise no major activity or tasks for developers to do day to day

**Accessing AWS- Common tasks (DevOps)**: DevOps Engineers have more to do in AWS through the Rackspace SLA. They need to manipulate Autoscaling Groups (e.g. Based on Dyson analytics and AWS cloudwatch, whenever a peak campaign is known to be coming so prescale rather than autoscale). Ordering or creating new/resizing infrastructure involves ourselves autoscaling groups, loadbalancers, EC2 instances etc. generally we have always ordered this via tickets in Rackspace but increasingly we are creating things. E.g. new Market coming online, or recently we needed another RDS in staging. Monitoring and performing patching and updates e.g. for the RDS cluster the patches available will be gets notified by AWS console. Config and admin of SSL and SES where SES is an amazon messaging service so we use SES for all transactional emails. Maintaining and pushing up to date ssh-keys for all developers on all AWS EC2s (utilizing dyson-vagrant and a separate ansible script held in secure repo) to ensure PCI compliance of environments. The management of those SSH keys can be done by Dyson vagrant not by AWS console in another repository.

**Manipulating Autoscaling groups**: We have Cloud Formation templates for the autoscaling groups but things like min/max instances are controlled via parameters.

* Go to CloudFormation then Filter/Search (e.g. “ASG” or “AU”)
* Choose the Stack of the autoscaling group you want to manipulate
* Click Update, “Use current template”, and click Next
* Generally we don’t touch anything in here other than “Maximum instances” and “Minimum Instances”. Most markets run 2 minimum and max 12, KR and AU run 4 minimum.
* If AU was running a big campaign we might up the minimum instances to 6 or more. obviously autoscaling will kick in, but often Dyson campaigns surge or traffic to the point that autoscaling just isn’t quick enough
* This was a problem pre-akamai, we’ve not seen (or perhaps noticed) a major surge campaign since Akamai so autoscaling slowness might not be a problem going forward.

**Ordering New Infrastructure within AWS**: For each market, we are requesting new loadbalancers at frontend, new autoscale groups, new EC2 instances, and new autorecovery. Redis and RDS are the shared resources, they don’t normally increase as markets are added to platform (Basically, the limitation for Redis is 15 databases and Magento needs 3 per Market). This is initiated via a Rackspace ticket, with a description of the new resources required in the body of the ticket following a preset template. Resolution times are usually unpredictable and unsatisfactory (it can take up to 1 week due to different teams within Rackspace working on it) and there are nearly always mistakes made due to the nature of the request. New infrastructure is usually just per-market resources and not the shared components.

**Ordering New Infrastructure Rackspace Ticket:** We request that Rackspace does the following in either staging and/or prod (depending on the requirement):

● Create new per-market CloudFormation stacks (based off an existing stack as an example)

● Change shared instances (NFS and FTP are shared resources in Backend) to add new folders and SFTP accounts

● Create New Frontend EC2s in an AutoScalingGroup (that mount the NFS backend for shared files)

● Create a New AutoRecovery Admin Instance

● Create New ElasticLoadBalancers for the Frontend ASG and admin instance

● Change existing shared SecurityGroups to allow access to/from new EC2s and shared backend services (RDS, REDIS, NFS and FTP). This should be done by editing and re-running the original Cloud Formation (CFN) templates that created shared resources, but there is a lot of drift caused by manual security group changes, so this step is done manually.

**Ordering New Infrastructure (Environment Validation):** Once Rackspace have finished setting up the new CFN stacks and AWS resources, DevOps has to validate they did it successfully and didn’t miss/overlook any steps or details. SSH into each new EC2 instance of Dyson vagrant to check NFS mounts are correct. Use dyson-cli commands provision and site:setup to configure the new environment/instances. Manually import a copy of the market (MySQL) database into RDS (the current dyson-cli deploy process assumes an existing release is already installed and running and can’t automatically deploy “from fresh”). Deploy the market-specific magento using the dyson-cli deploy command, to test a full deployment (and AMI frontend creation) occurs without errors. Check all filesystem permissions and ownership is correct (especially for shared files). Add CloudFormation Event Rules to enable RDS backups. Request DNS changes (from Accenture) to validate new SSL certificates for Load balancers, then attach the SSL certificates to load balancers (more details in a later slide). Request DNS/Akamai changes (from Accenture) to point traffic at Load Balancers (working with project managers as part of go-live cutover plan).

**Monitoring and Performing Patches/Updates:** CloudWatch monitoring, metrics and dashboards are in place for all production environments including CPU, RAM usage etc. Alerts are setup for NFS disk, CPU and RAM usage >90%. We use SNS to pick up the alerts and then email to G&V staff (Developers and DevOps). We also integrate those alerts through webhooks into pagerduty so an alert is sent via PagerDuty to staff on Out Of Hours (OOH). DevOps regularly manually reviews CloudWatch graphs to spot any odd CPU or network traffic trends (to check any alerts aren’t triggering). Ubuntu patches are manually applied by DevOps on a quarterly basis (and within 30 days for critical patches) utilising the load balanced frontends, We can temporarily take one frontend from the loadbalancer and will apply the patch and the public website is available as the other frontend will allow the traffic for the patched frontend and we continue the same process for the other frontend so there is no downtime of shop fronts for customers. RDS and REDIS are in Writer/Reader clusters of backend so can be resized/patched as needed without downtime. Shared EC2 instances (NFS, FTP and Bastions) need scheduled downtime during patching, which impacts other services that rely on them.

**Managing SSL via Certificate Manager:** All of our certificates were issued by AWS Certificate Manager. There are actually two certificates will play here, one of them is dyson.com certificate issued by GeoTrust. This is an SSL lives on Akamai and is completely out of our control. This certificate gets done or approved by Accenture and that can be auto renewed or manually expired and might get notification, all these things were out of control. Even though Dyson are providing SSL at an Akamai level, which is what the customer sees... We still need to provide SSL at the load balancer level, and we manage and issue these certificates in AWS Certificate Manager. Certificates are always DNS backed, not email, otherwise they will fail to auto-renew. Create Certificate, choose DNS validation, and the DNS records will be generated which need sending to Accenture. Remove the \_ prefix from the values before sending.

* DNS Name: \_820987380d8b3225cd2be74c12d630c1.staging.dyson.com.ro.
* DNS Record Type: CNAME
* DNS Value: 8918330c053bd5a18fb312fed0daf934.vtqfhvjlcp.acm-validations.aws.

DNS validation records are only valid for 72 hours so it is advised to book Accenture in to do the work, The key thing here is we need to provide the booking date/time and make it confirmed when the ticket has been actioned and we can stress the 72 hour window to gets it completed in Accenture sprints, in order to commit and then create the certificate request/DNS records, otherwise by the time it is actioned the DNS records have expired. If DNS record have never deleted, then every year DNS records will silently renew with no further action needed.

**Managing Email Addresses in SES:** SES is kind of Amazon transactional messaging service. We use AWS SES to send pretty much older transactional email for most of the Magento 2 sites. Note some SAP CRM markets may send email instead of using SES (e.g. Dyson Mexico market). Mostly all the markets of Magento will sent emails through SES. We only have valid running SES in “Dyson EU Production” account – the other 3 AWS accounts will rely on same SES account used by the “Dyson EU Production” so KR email addresses need to be sorted in the Dyson EU Prod Account. In SES we can validate in two ways of sending email, One way of sending emails is we occasionally in the past done full SPF type of authentication on the domain at a main level allowing us to send from anything@domain.com. The other way of doing it is we have been validating individual email addresses. The email address you want to validate needs to be a real inbox that someone can receive incoming emails to (to click a link and validate the address in SES). Add the email address into SES and submit. Once the email instructions have been followed we should be able to send from that email address.

Once an email is approved in SES it needs to be added in a few places to work. Magento store email addresses in Stores > Configuration > General > Store Email Addresses. In Ansible’s Group Vars, Ansible will deploy Postfix config in the middle that relays to SES. Basically postfix connects to the SES, if it connects fine then we’ll get notifications from email. Those postfix logs are monitored with help of Amazon cloudwatch. Ansible will deploy an encrypted credentials file for SES provided by Rackspace. It will also add the email address from the group vars into the Postfix config. The main listed email address is also stored in ansible group vars (from email address: [xyz@mail.com](mailto:xyz@mail.com) in ansible group vars folder). If we have provided the email address in Store Email Addresses configuration and you haven’t put it in Ansible group vars which is deployed in postfix config then it results in failing to authenticate with SES. If it’s a valid email address in Ansible group vars and provided invalid one in Store Email Addresses config then you will be connecting to SES then it’ll bounce back and fails as it doesn’t recognize the email address in Magento store. If we provided a right credentials file, right email address in group vars of post fix config and right email address in Magento store. If all the mentioned three were in right place then SES will issue an alert. This allows Postfix initially to connect to SES. Magento will dictate the actual send from email, so they just need to match what you got approved in SES. If the market changes the store email addresses in Magento, they will break their email sending.

**AWS Inventory Naming and Tags:** Things in AWS will generally be named in a consistent way using the country code of the market they are related to. This is mainly to make things easy to navigate and read. E.g. Australian (AU) market resources can be named as:

* Load balancer called Magento-AU, the admin for Australian market is called Magento-AU admin.
* EC2s called Magento-AU
* Security groups called sg-PrivateWebAU (internal VPC traffic to the shopfront), sg-PublicWebAU (public traffic from the web).

There are some custom EC2 tags that are critical though:

* EC2 Tag “Site”: must be set to the country code e.g. AU
* EC2 Tag “Site Type”: must be set to be “Frontend” or “Admin” based on which ever it is

If these naming’s and tags are missed, the Dyson CLI cannot find what it is looking for.

**AWS Automated Tasks / Lambdas**: We have a few lambdas run and these were configured in environmental setup. Each region (staging/prod eu/kr) has its own copy of lambda functions.

* Dyson\_siteDetails - Triggered by dyson-cli (ansible) and CloudWatch Events. Routes commands through to other lambdas
* Dyson\_amiCleanup – It’ll run at the end of deploy to remove old AMIs after a new frontend AMI is created.
* Dyson\_amiPublish - publishes a new AMI and gives AMI id to CFN stacks after it's created it’ll spin up the new instances based on the new AMI. Triggered after a deploy or by dyson-cli
* Dyson\_databaseBackup - SSHs to the admin instance for a market and backs up RDS database (using magerun script) to S3 using n98-magerun db:dump command. This can also be manually triggered by using Dyson CLI command Dyson site:backup.
* Dyson\_warmCache - triggered at the end of deploy to crawl URLs. Expects all URLs for a site, but is only supplied with the homepage.

Lambdas use a shared ssh-key in an S3 bucket for shell access to admin instances. CloudWatch Event Rules trigger DatabaseBackup Lambdas every 12 hours. Lambdas are initially deployed to AWS by dyson-vagrant/cli per region.

**AWS Backup**: AWS backup runs once daily (overnight around 11pm UTC) on instances with Tag: Backup=True. We got 20 days retention on these backups. Snapshot backups taken of all admin EC2s and shared NFS, FTP and Bastion EC2 instances. Snapshot backups taken of all RDS and REDIS clusters. We do not backup running frontend instances because they are automatically created from AMIs by ASGs and CFN. They are considered disposable and can be easily recreated from the last known good AMI. Per-market mysql backups (taken every 12 hours and just before deploy) created by the Dyson\_databaseBackup lambda are retained for 60 days in S3 buckets.

**Accessing Digital Ocean and Common Tasks**: Because there is just one Digital Ocean Droplet, there are very few reasons to go administer the Droplet via the Digital Ocean web interface. The droplet is fully integrated with the Dyson Vagrant as it does a lot of configuration and all of the ansible is coded so if it knows it’s targeting Digital ocean rather than AWS then different of tasks gets triggered during playbook execution and it’ll exclude things related to RDS and it’ll use only local MySQL database and Redis. Droplet is provisioned by it. Nginx, Site setup and postfix is configured by it. Deploys run as you’d expect to it. Free disk space is low, but migrating to a newer (bigger) droplet will allow this to double from 160 GB to 320 GB. If we lost the droplet for some reason and we need to build a new one then we would open the droplet and we will be having snapshots to build it. If we don’t have snapshots as well then there’re Readme.md files inside the Dyson vagrant folder so with help of those files we create a new droplet and SSH into it then setup a new special deploy user and add it to the sudo user’s list. Provide the IP address in Ansible inventory and won’t login to Digital Ocean again because the Dyson vagrant will provision it. site:setup will configure Maria dB, Nginx and Postfix and finally it deploy.

**Migrating the Droplet**: Dyson will need to provide a new Digital Ocean account with valid billing details attached. Dyson need to issue G&V with a technical account on their Digital Ocean so that we can create droplets we need and manage droplet snapshots in advance and this will gives us droplet IP address. G&V creates new Droplet and puts in DNS request to Accenture for all new pre-prod subdomains. I.e. to replace krdyson.gandvclients.co.uk with preprod.dyson.co.kr. Need some non-regional DNS records for gold/sandbox/demo sites (use \*.m2.dyson.co.uk? where \* refers to gold/sandbox/demo sites). The first thing we need to do is content freeze droplet onto the G&V site and take a snapshot results in a notification going out to Dyson content teams that any markets are using it. If you work on these site now you may probably lose any changes you’ll do because we’re taking the backup of migrating droplet and is basically offline for 24 hours during backup time. We will send snapshot to Dyson Digital Ocean Account so that there we can transfer these snapshots from account to account. At some point we need to restore new Dyson droplet from snapshot ($80 PCM droplet: 8 CPU cores, 320 GB storage) and it will keep the same IP address. Update Ansible Group vars for all preprod sites with new domains and put new IP address in staging inventory. Digital ocean make use of Ansible inventories because there is only static IP address and AWS won’t be using Ansible inventories as there were dynamic inventory and we have to use EC2 API extension to go through and request the dynamic inventories. Rerun site:setup for every market. We need to manually go through the Digital Ocean droplet and update domains in config table in Magento. Check Letsencrypt config/certs working and Smoke test need to be done at each pre-prod site.