


[SOP] Access Standard Operating Procedures

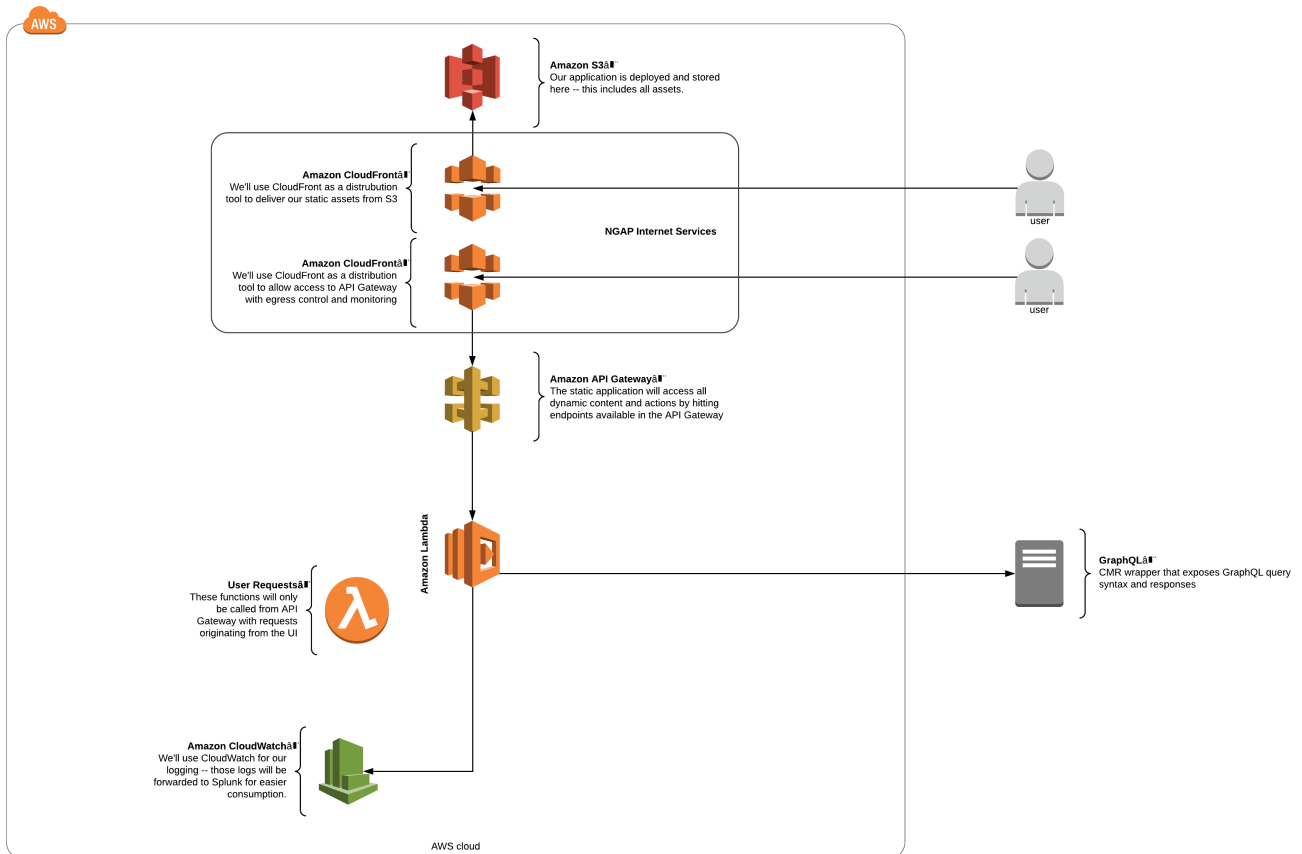
 Work in progress

Earthdata Access (<https://access.earthdata.nasa.gov>) is a 508 compliant application that allows users to search, discover, and access NASA Earth Observation data. Additionally, Access will offer a feature that allows other applications (like CMR and MMT) to render collection metadata pages that surfaces helpful information like service associations and related collections.

Environment

- **Production**
 - Public Facing
 - <https://access.earthdata.nasa.gov/>
 - esdis-app-edsc-prod-7557 [NGAP 2.0]
- **UAT**
 - Public Facing
 - <https://access.uat.earthdata.nasa.gov/>
 - esdis-application-edsc-uat-8982 [NGAP 2.0]
- **SIT**
 - *VPN only*
 - <https://access.sit.earthdata.nasa.gov/>
 - esdis-application-edsc-sit-8022 [NGAP 2.0]

Architecture



Components

- Earthdata Access is built using Node JS, React and the Serverless framework

- GraphQL which sits on top of CMR and allows for an improved data retrieval experience.
- Earthdata Login Single-sign-on (external component dependency)

Deploying

Generally, building and deploying will be done from Bamboo. Bamboo's deployment mechanisms give the option of a script defined within the UI or a script within the repository alongside the code. Earthdata Access uses a script that is defined within the repository located at [bin/deploy-bamboo.sh](https://github.com/earthdata/earthdata-access/blob/master/bin/deploy-bamboo.sh). The script is simply a list of commands so if Bamboo isn't an option, the commands can be ran manually via the command line. The overall process looks like this:

1. Install necessary libraries
2. Build static assets
3. Deploy Infrastructure
 - a. Roles
4. Deploy Application Resources
 - a. Lambdas
 - b. CloudWatch Events
 - c. API Gateway
 - d. CloudWatch Logs
5. Deploy the static assets to S3

Required Environment Variables For Deployments

- **API_HOST** The API Gateway endpoint created during deployment. For NGAP deployments this will be a CloudFront endpoint that points to the created API Gateway.
- **APPLICATION_HOST** URL of this application once deployed, used for linking back to the application and OAuth.
- **AWS_ACCESS_KEY_ID** AWS Access Key created in CloudTamer.
- **AWS_SECRET_ACCESS_KEY** Access Token created in CloudTamer.
- **CLOUDFRONT_BUCKET_NAME** NGAP dumps CloudFront to a bucket because we don't have permission to them, we use our Lambda, CloudfrontToCloudwatch, to read these logs and send them to Splunk.
- **EDL_HOST** URL of the EDL API to use for authentication.
- **FEEDBACK_APP** ID of the Earthdata Feedback App to supply to TopHat.
- **GRAPHQL_HOST** URL of this GraphQL application used as the datasource for this application.
- **GTM_ID** Google Tag Manager ID.
- **LAMBDA_TIMEOUT** Value set as each Lambdas timeout, a handful of Lambdas have custom timeouts for specific reasons.
- **LOG_DESTINATION_ARN** AWS ARN of the log subscription provided by NGAP.
- **STAGE_NAME** What label to apply to the deployment, helps separate multiple deployments within a single account. Earthdata Access uses the standard EED environments 'sit', 'uat' and 'prod'.
- **SUBNET_ID_A** AWS Subnet ID provided by NGAP.
- **SUBNET_ID_B** AWS Subnet ID provided by NGAP.
- **VPC_ID** AWS VPC ID provided by NGAP.

Destroying

This can be done from within the AWS console by deleting the stack from CloudFormation. You may experience issues related to an S3 bucket not being empty, you can empty the bucket and continue deleting assets if you experience this. Running the commands require valid access tokens, see the Serverless Framework documentation for methods of providing these values.

Using AWS Access Tokens (Serverless Framework): <https://www.serverless.com/framework/docs/providers/aws/guide/credentials#using-aws-access-keys>

```
// Destroy Application resources and static content
serverless destroy --stage sit

// Destroy database and application roles
serverless destroy --stage sit --config serverless-infrastructure.yml
```

Configuration

Earthdata Access is deployed to NGAP 2.0, aka AWS using the [Serverless Framework](#).

Backup

Code Backup

Earthdata Accesses code base and static assets are stored in BitBucket. BitBucket is backed up on-premise as part of EED-2 infrastructure management.

Bitbucket: <https://git.earthdata.nasa.gov/projects/EDSC/repos/edsc-cmr-preview/browse>

Restoring (Deployments from Scratch)

Should Earthdata Access need to be completely recovered (static assets and Lambdas), that process takes around 30 minutes and requires a few tickets that requires additional resources (NGAP).

Tickets depending on other teams:

- AWS CloudFront endpoint for S3 bucket
- AWS CloudFront endpoint for API Gateway

Once these tickets are completed:

- Output from the tickets (S3 bucket and API Gateway endpoint) need to be set within Bamboo.

Steps:

1. Create the deployment bucket on AWS (`earthdata-access-[ENV]`)
2. Deploy full application
3. Configure CloudFront custom error pages, documented here: [ReactJS Static Website Hosting](#)
4. Create the tickets above, the outputs from the deployment should be provided in the tickets (s3 bucket name and API Gateway ID)
5. When tickets are completed, the values will need to be saved in Bamboo.
6. Deploy again so that the ENV variables are provided to the lambdas.

Ensure that all environment values are provided, when deploying from Bamboo those values are set in the UI, but when deploying from command line they need to be provided or set.

Release Cycle

Earthdata Access follows the SAFe process as implemented by EED. Typically, that means we plan priorities in 3 month increments and release code every 2 weeks. If needed, Earthdata Access can release on-demand with appropriate notice to stakeholders.

Patching and Remediation

Earthdata Access has a blocking step in our deployment process that audits our libraries and dependencies. Once a vulnerability is found, steps are taken to patch and update and resolve the vulnerability immediately.

In the case of a vulnerability discovered for a resource currently deployed, a ticket is filed once the issue is identified. Earthdata Access devops consults with the security team to prioritize the remediation of the vulnerability found. Once a ticket is created, approved, and prioritized, Access dev team works the issue until all vulnerabilities are resolved and deploy the updated app to all operational environments.

Diagnostics

Earthdata Access logs to AWS CloudWatch on a per Lambda basis; this allows you for easy access to specific Lambda logging in the event that you know which Lambda is responsible for the logs you're looking for. If a wider search needs to occur, Earthdata Access forwards logs to Splunk which accommodates a wider array of search abilities.

Splunk: <https://logs.earthdata.nasa.gov/>

Planned Maintenance

Deployments are handled via Bamboo.

Bamboo: <https://ci.earthdata.nasa.gov/deploy/viewDeploymentProjectEnvironments.action?id=374046723>