AWS Cloud Visualization Tool (**V-Edda**)

Introduction

One of the challenges for administrators, architects and developers in the cloud environment is to comprehend the relationships between components deployed in the cloud. Most management consoles do not clearly indicate the contextual relationship between components and this often leads to the engineers or the admins to get redundant components up, make changes that may not be in line with the overall architecture vision etc.

The goal of this project is to provide a new visualization tool for AWS such that the contextual relationships between components can be clearly identified and explained. Some of the queries that can be enabled via this tool that may not be clearly understood via management consoles are:

1. How many volumes is an instance associated with?
2. Which instances are open to public i.e. has a public IP?
3. Are all my instances covered by the right security groups within the VPC? If not can I detect which ones are open and where the security hole is?
4. What subnets are associated with my deployment? Are all instances in their own subnet? Does the subnet ensure segmentation and isolation to enhance security?
5. What availability zones are my instances placed in? For a given VPC how can I easily get the distribution?
6. How many AMIs are available for this setup? Is it possible to reduce redundancy and maybe have two or more instances use the same image to reduce management effort?

The work can be easily expanded to support various other use cases as long as it can be supported by the backend infrastructure.

Below is a description of some of the design and implementation details for this tool which is currently named **V-Edda** short for Visual EDDA. More details on EDDA are provided below.

Design

Backend Infrastructure:

As the name indicates, this tool provides the front end for EDDA which is a cloud setup monitoring tool from Netflix. This tool periodically polls cloud components to record changes over time. It provides a great platform for querying information that requires prior knowledge regarding contextual information. EDDA returns basic JSON responses and has extensive support for queries like providing callbacks for JSONP etc.

However EDDA is still a work in progress and this was only recently open sourced by Netflix. This tool was developed to query the existing supported EDDA APIs. EDDA works by region i.e. an instance of EDDA can poll only one region. We could have multiple instances of EDDA running to poll as many regions as required.

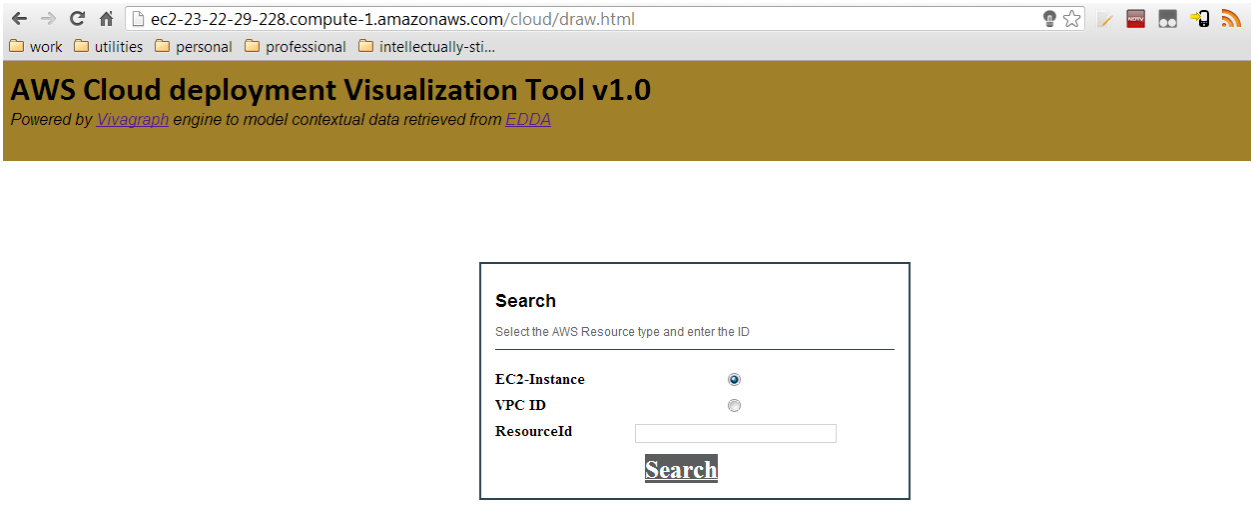
Please see below for accurate deployment information and how to use EDDA within Symantec.

User Interface:

Querying EDDA can be cumbersome as specifying a suitable search matrix is complex. This tool plans to alleviate that by providing a simple user interface where users would be able to filter results based on set parameters. This in turn is translated into a search matrix as a part of the API request to EDDA.

EDDA's responses are JSON formatted and often complex depending on the kind of search query we provide. One of the highlight of this tool is that it provides a graphical depiction of the deployment environment in the sense that we get a graph of nodes and edges denoting different cloud components and how they are contextually related.

Currently you can query this tool for information regarding a VPC or an EC2-Instance(s).

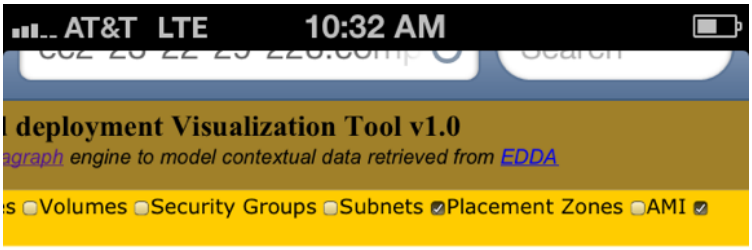


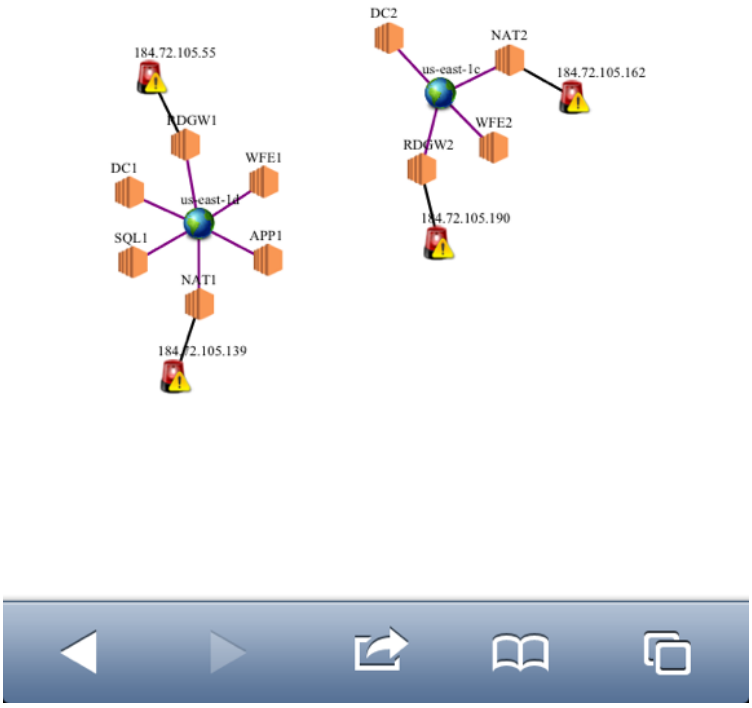
This feature can be extended in future to support querying for other components.

The response is provided in the form of graphs. Towards this end an advanced graph engine called Vivagraph was used. This is built entirely on javascript platform and supported by Firefox, Safari and Chrome.

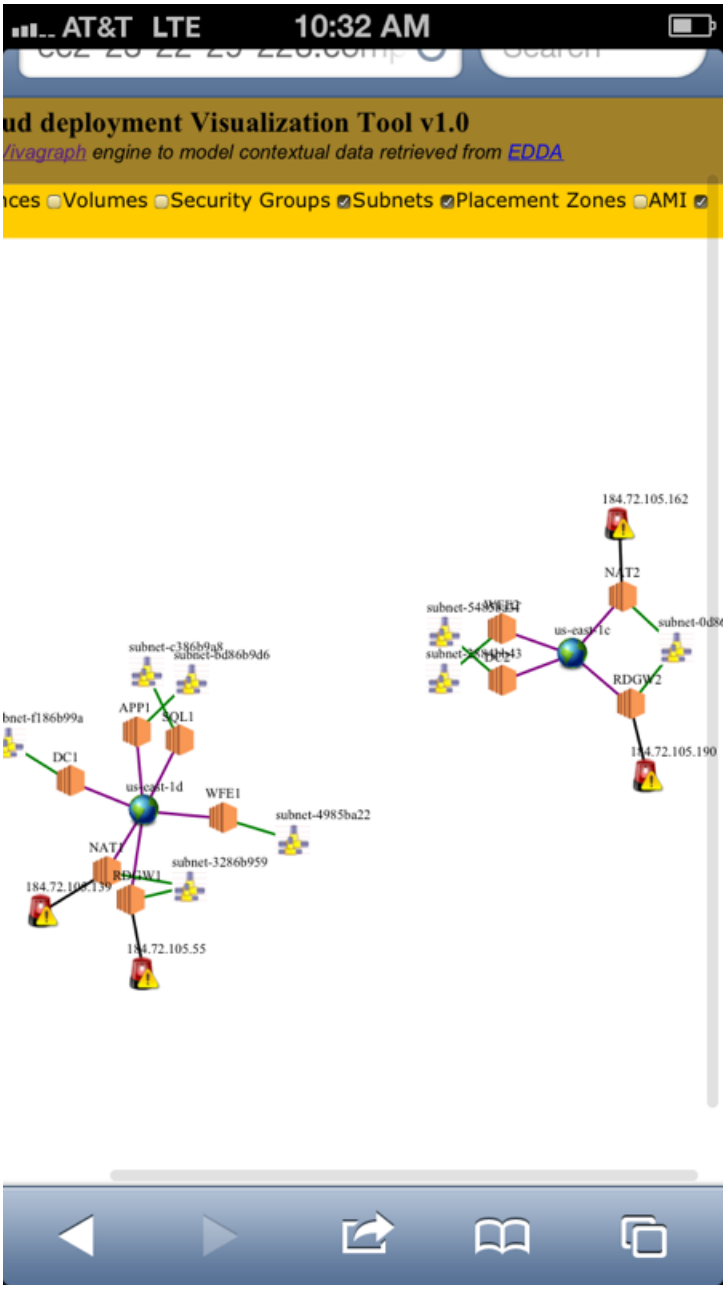
Below are some of the screenshots from a smartphone denoting the different ways we can query cloud deployments.

1. Use Case: Show **placement zones** for instances **in vpc-7886b913** including those instances that have a **public IP address**:

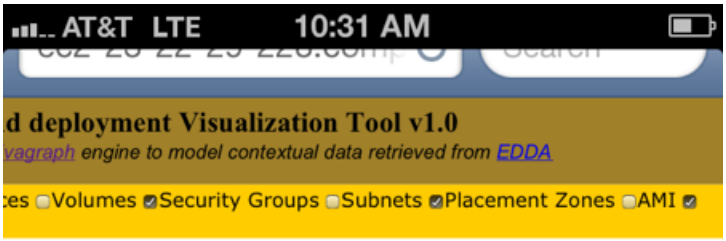


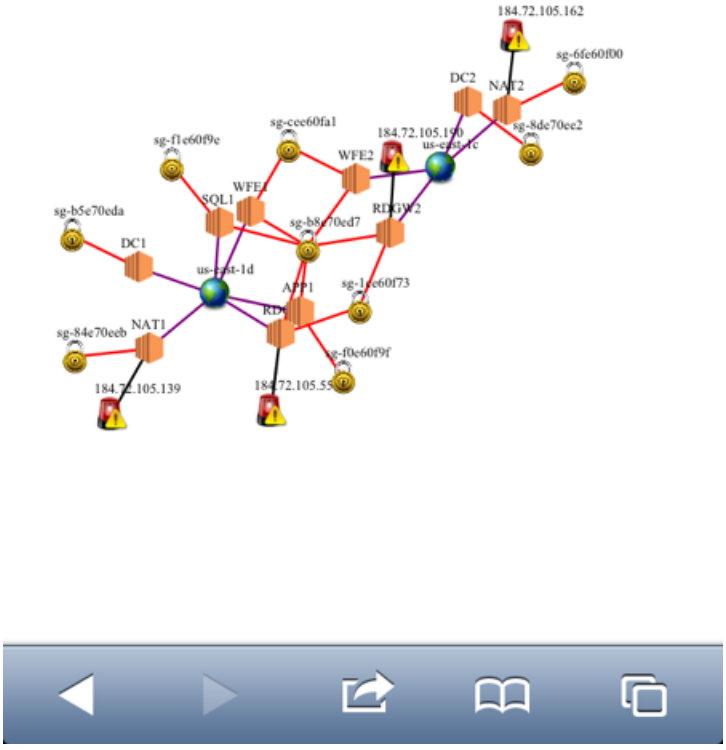


2. Use Case: Show **placement zones** and **subnets** for instances in **vpc-7886b913** including those instances that have a **public IP address**:

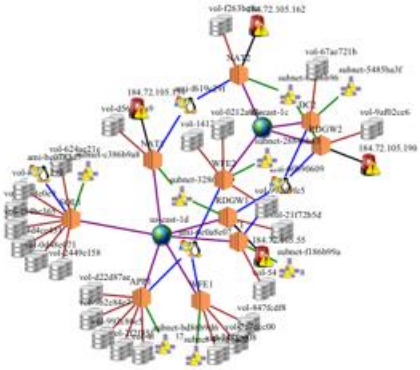


3. Use Case: Show **placement zones** and **security groups** for instances in **vpc-7886b913** including those instances that have a **public IP address**:





4. Use Case: Show **placement zones, subnets, volumes, AMIs** for instances in **vpc-7886b913** including those instances that have a **public IP address**:



The above use cases highlight some of the common queries we can make using this tool. The interface is simple as checkboxes are used to filter the desired components.

The screenshots above are small as they are captured on phone. The challenges section below goes over some of the issues faced in developing this tool.

Implementation

This primarily involved working with javascript to modify the powerful vivagraph platform to suit our needs.

Some of the high level steps in implementing the tool are documented below:

1. Creating basic template to include overlay query dialog.
2. Create handlers for checkboxes to apply filters to graph.
3. Add Resource ID to node object in the graph library to denote node information like subnet – subnet-0d86b966 etc. over the node.
4. Color code edges linking instances and related components to be able to easily differentiate between different relationships. Refer **Link color coding** scheme below.
5. Include icons that clearly indicate the cloud resource for node elements. Refer **Cloud Components Icon assignment** scheme below.
6. Handle asynchronous nature of ajax calls when querying for various instances. This operation needs to be **synchronous**.
7. Handle error responses and setup logging. (In process)

Link color coding

Brown – Link between an EC2-Instance and its associated Volume(s)

Red – Link between an EC2-Instance and its associated Security Group(s)

Green – Link between an EC2-Instance and its associated Subnet(s)

Purple – Link between an EC2-Instance and its associated Placement Zone

Blue – Link between an EC2-Instance and its associated AMI

Black – Link between an EC2-Instance and its associated Elastic IP(s)

Cloud Components Icon assignment

 - A single EC2 Volume associated to a single Instance if any.

 - Denotes a single EC2 Instance

 - Denotes a single EC2 Security Group bound to an Instance, multiple Instances or no Instance

 - Denotes a single subnet configuration bound to an Instance, multiple Instances or no Instance

 - Denotes an AMI image that is associated with an Instance, multiple Instances or no Instance

 - Denotes a placement zone associated with one or more Instances

 - Denotes a public facing instance having an Elastic IP address

 - Denotes a Load Balancer that is associated with one or more Instances

Challenges

Some of the challenges faced are enumerated below:

1. EDDA is not mature in terms of configuration and flexibility. There were numerous interactions with the Netflix engineers about configuration and stability issues.
2. EDDA is not configurable to be deployed on a port different from 8080. This port is blocked internally by Symantec LAN on the inbound. Thus for any testing to be done, it would have to be done outside the Symantec LAN.
3. EDDA process is not stable when it polls a region different from the region it is deployed in. This issue has been filed with Netflix engineers.
4. Vivagraph engine is mainly catered to do UI design with simple templates that contain nodes and edges. It is not flexible to provide per link/edge level control to be able to get different lengths for different components or based on different weights. This engine expects a strict one time definition for all nodes and edges and this instance is used.
5. Vivagraph uses WEB-GL extensively and there some issues in determining node definitions to be suitable for our use cases.
6. EDDA does not provide adequate information regarding the exact operating system version in an AMI. This causes difficulty in determining nature of applications running and hard to do any sort of app introspection as we would not which agents to install on the system.
7. EDDA is still evolving in terms of API support so we are still bound by the number of things we can query.
8. EDDA does not provide any authentication support so we cannot expose this endpoint to the world.
9. EDDA provides JSONP support required for cross domain requests. However it's **imperative** to craft Ajax requests such that the callback is specified first before the search parameters. Sample API URL:
   *http://<server>:<port>/edda/api/v2/view/instances/<instanceId>;**_callback=callme**:(imageId,securityGroups,subnetId,placement:(availabilityZone),tags:(key,value),publicIpAddress,blockDeviceMappings:(ebs:(volumeId)))*

Deployment Information

1. **EDDA**:
   Instance Name: ec2-174-129-62-228.compute-1.amazonaws.com
   Port: 8080
   Instance key: debskey
   Server Location: */opt/edda/*
2. **V-EDDA – Visualization Tool**:
   Instance Name: ec2-23-22-29-228.compute-1.amazonaws.com
   Port: 80
   Instance key: debskey
   Server Location: */var/www/lighttpd/cloud*

To try out EDDA and V-EDDA or if in need of more information please contact me at Krishnan_Narayan@Symantec.com. I'd be happy to show a full demo as well.