High Availability Networking in AWS VPC
By: Warren Turkal
2015.03.16
Who am I?

• Infrastructure Hacker at SignalFx
• 12+ years experience in tech
• 3+ years hacking on AWS
What I'll Be Talking About

- Traditional AWS solution
- My requirements
- Details of my solution
- Additional benefits
Traditional VPC Design Overview

Inside single region

VPC

Private Subnet (Inside single AZ)

- server instance

Public Subnet (Inside single AZ)

- NAT router Instance

igw
Problems with the Traditional Design

• NAT router is a SPOF
• No quick failover for NAT router failure
• No layer-3 networking between hosts in different regions
My Design Criteria

• Provider agnostic abstraction
• Abstraction boundaries reflect the cost/performance/reliability boundaries inside AWS and other cloud providers
• Isolatable zones of maintenance
• Enable developers to self serve infrastructure (e.g. hosts and load balancers)
• Work within our current processes for software deployment
• Allow experimentation
The Onion of Abstractions

Region – AWS Region (e.g. “aws-us-east-1”)

Culture – AWS Virtual Private Cluster (e.g. “aa”)

Cell – AWS Availability Zone (e.g. “aaaa”)

Hosts – AWS EC2 instance (e.g. webserver1--bbaa)
The Onion of Abstractions

Region – AWS Region (e.g. “aws-us-east-1”)

Culture – AWS Virtual Private Cluster (e.g. “aa”)

Cell – AWS Availability Zone (e.g. “aaaa”)

Hosts – AWS EC2 instance (e.g. webserver1--bbaa)
Cell Design

Cell “aaaa”

Inside single AZ (e.g. us-east-1a)

Routing Subnet

AWS Network

Private Instance Subnet

Frontend Subnet
Culture Design

Culture “aa”

Cell “aaaa”  Cell “aaab”
Cell Routing and Culture Wiring

Cell “aaaa”
- Routting Subnet

Culture “aa”
- vpc
- igw

Cell “aaab”
- Routting Subnet
Regional/Global Design

Region "aws-us-east-1"
- Culture "aa"
- Culture "ab"

Region "aws-us-west-2"
- Culture "bb"
Interregion Highlevel Networking

Region “aws-us-east-1”

Culture “aa”
Cell “aaaa” Routers

Culture “ab”
Cell “abaa” Routers

Region “aws-us-west-2”

Culture “bb”
Cell “bbaa” Routers

Internet
Add a Corp VPN

Region “aws-us-east-1”
- Culture “aa”
  - Cell “aaaa” Routers
- Culture “ab”
  - Cell “abaa” Routers

Region “aws-us-west-2”
- Culture “bb”
  - Cell “bbaa” Routers

Corp VPN

Internet
Cell Internet Connectivity

Routing Subnet

igw

ER0

ER1

Private Subnet

host
Intracell Routing

Cell “aaaa”

Routing Subnet

Internet

ER0

IR0

IR1

ER1
Intercell Routing

Culture “aa”

Cell “aaaa” Routing subnet

IR0

IR1

Culture “bb”

“Cell bbaa” Routing Subnet

IR0

IR1

Internet
Intercell Routing (VPN tunnels)

Culture “aa”
- Cell “aaaa” Routing subnet
  - IR0
  - IR1

Culture “bb”
- “Cell bbaa” Routing Subnet
  - IR0
  - IR1
Eliminate ER Hop in Intercell Connectivity

Routing Subnet
- igw
- IR
- ER
- Other internal routes
- Everything else

Private Subnet
- host
• Cloudformation w/ custom scripting (python+troposphere lib+boto lib)

• Cultures and cells are managed with separate tools the rely on common lib

• Must keep track of dummy and intermediate addresses among other things for the config.

• Routing protocols are all implemented with Quagga.
Interesting Numbers

- A cell is composed of 86 AWS objects when running with 2 IR and 2 ER machines.
- A culture is composed of 22 AWS objects.
• We have cell-aware and culture-aware tools to spin up/terminate instances and load balancers (ELBs on AWS).
• These tools integrate with our Salt deployment to make it easier for developers to self-service when spinning up new types of instances.

Example of spinning up instance:
```
sfhost add webserver bbaa c3.xlarge
```

Example of spinning up an ELB:
```
Sflb --culture bb --name api create
```
SignalFx – Launched last week

**Streaming analytics on multi-dimensional metrics** for monitoring modern apps

Free 30 day trial

Say hi at our table for demo+discussion
Thank You!