Amazon Virtual Private Cloud (Amazon VPC) lets you provision a logically isolated section of the Amazon Web Services (AWS) cloud where you can launch AWS resources in a virtual data center that you define. In this session you learn how to leverage the VPC networking constructs to configure a highly available and secure virtual data center on AWS for your application. We cover best practices around choosing an IP range for your VPC, creating subnets, configuring routing, securing your VPC, establishing VPN connectivity, and much more. The session culminates in creating a highly available web application stack inside of VPC and testing its availability with Chaos Monkey.

Learning about High Availability Applications in VPC

- What is Amazon Virtual Private Cloud (VPC)?
- VPC common use cases
- VPC basics
- Why move to VPC?
- Connecting VPC with your data centers
- Making your VPC infrastructure highly available
- Making your application highly available

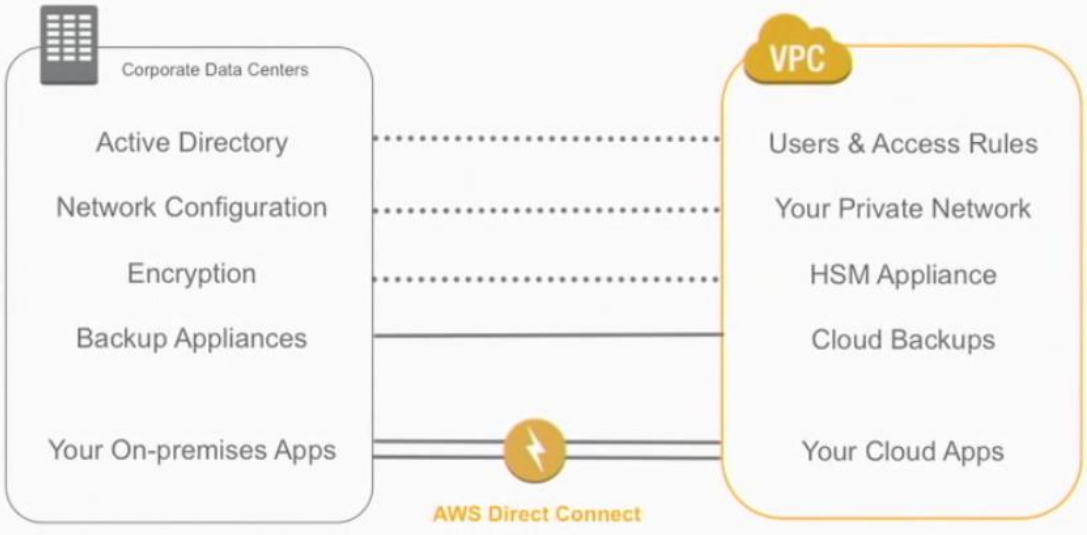
What is Amazon VPC?

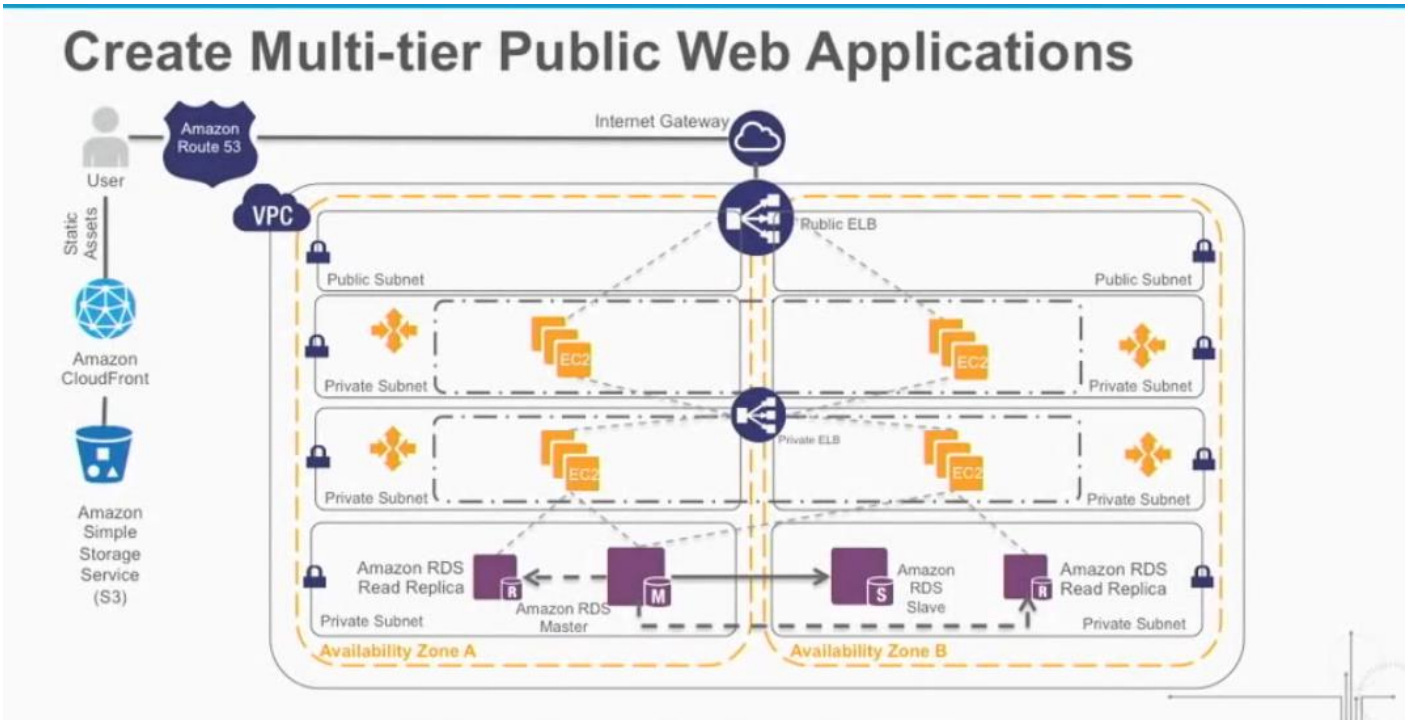
- A private, isolated section of the AWS cloud
- A virtual network topology you can deploy and customize
- Complete control of your networking

Most easily put, it is a virtual data center you can build out and control on AWS!

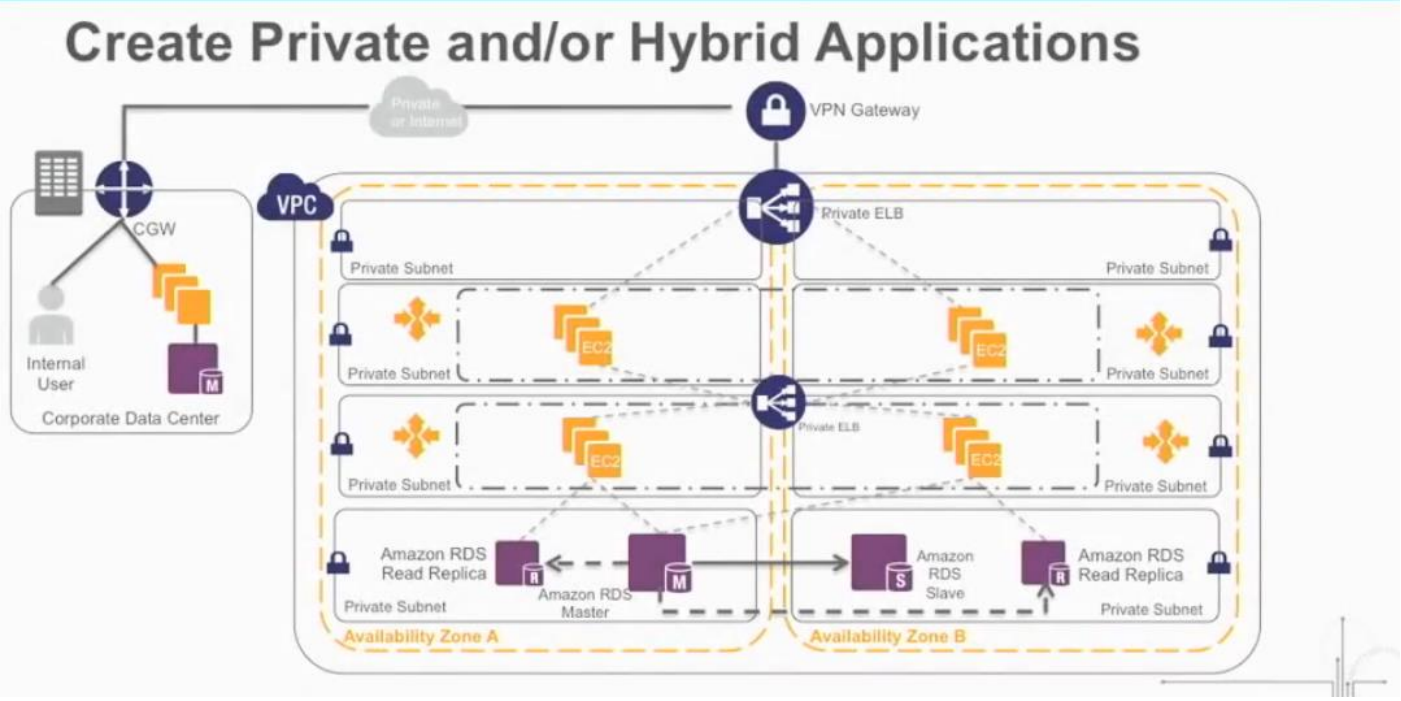
VPC Common Use Cases

Design a Virtual Data Center on AWS

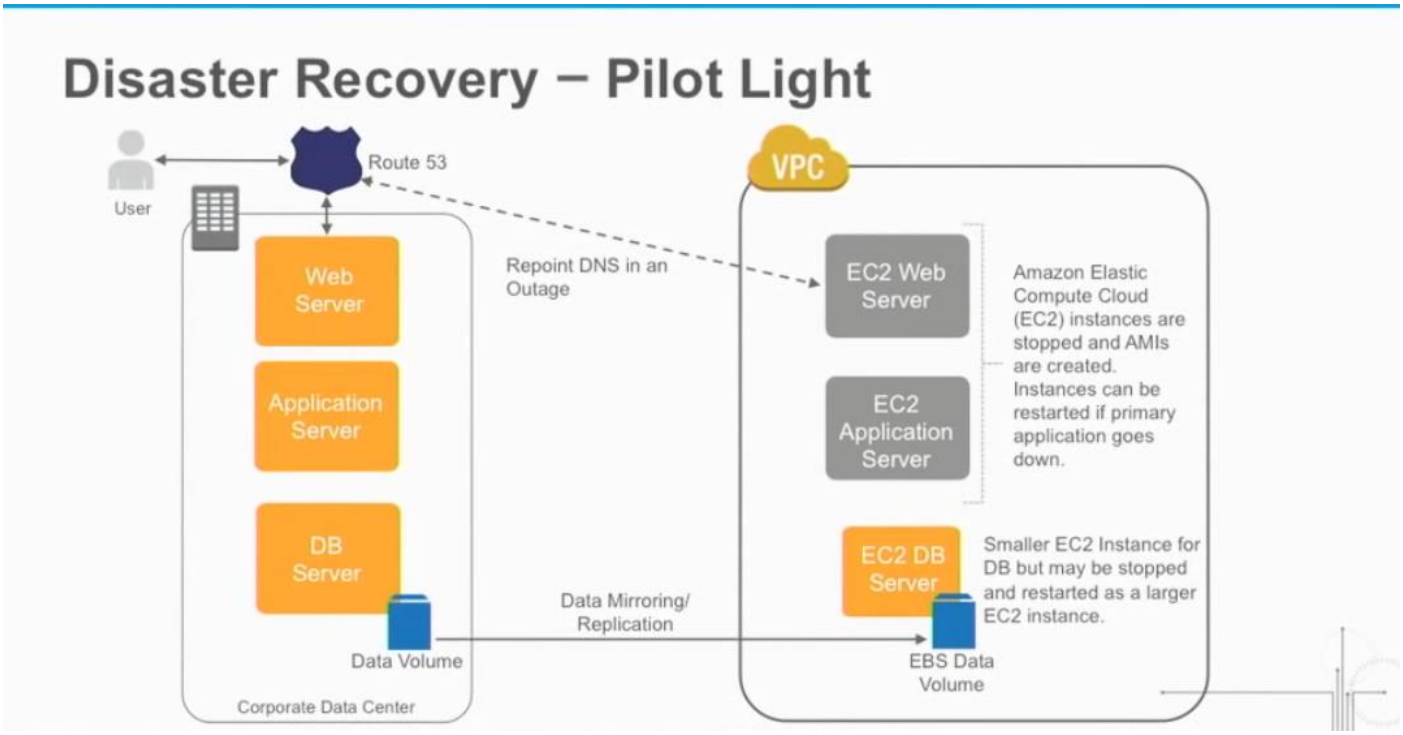




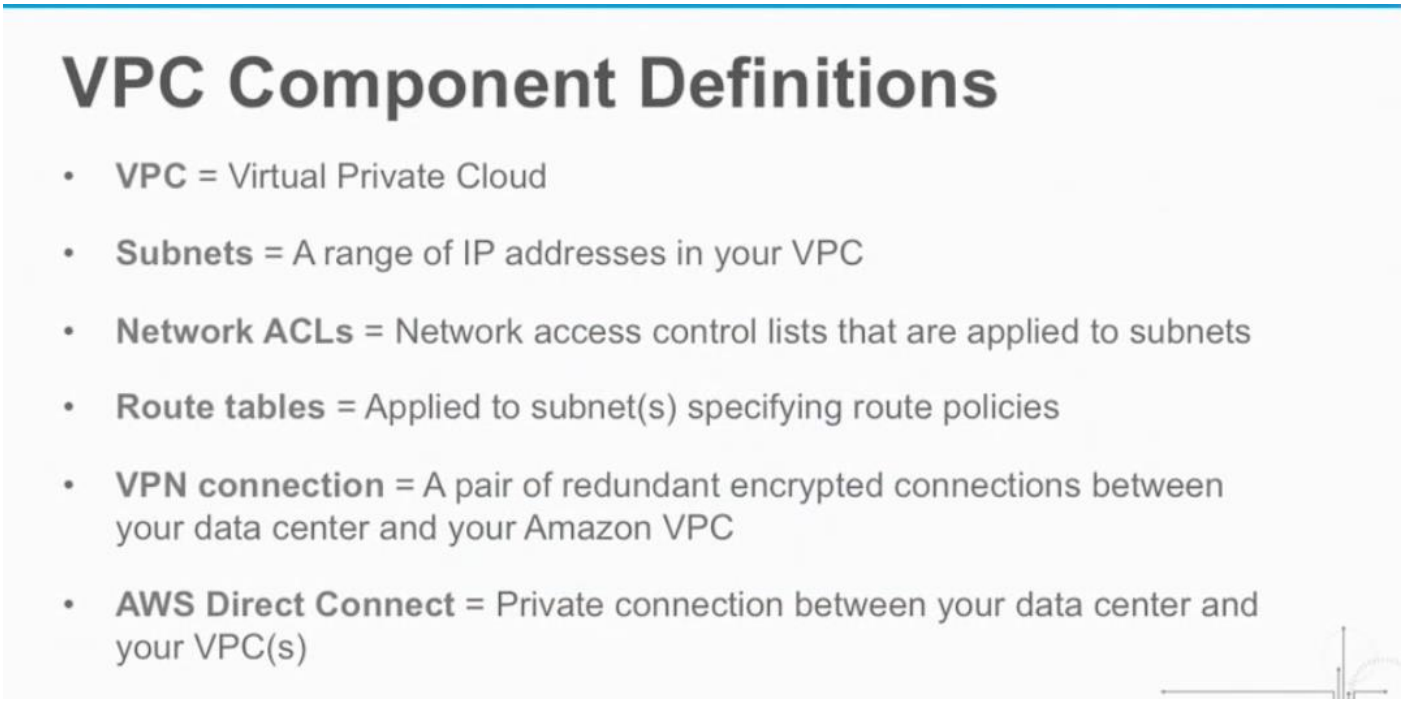
Notice that we are running a lot of our applications in private subnets, the only thing we are publicly exposing is our public load balancer. We also have a private LB that is only available between our web tier and our app tier.



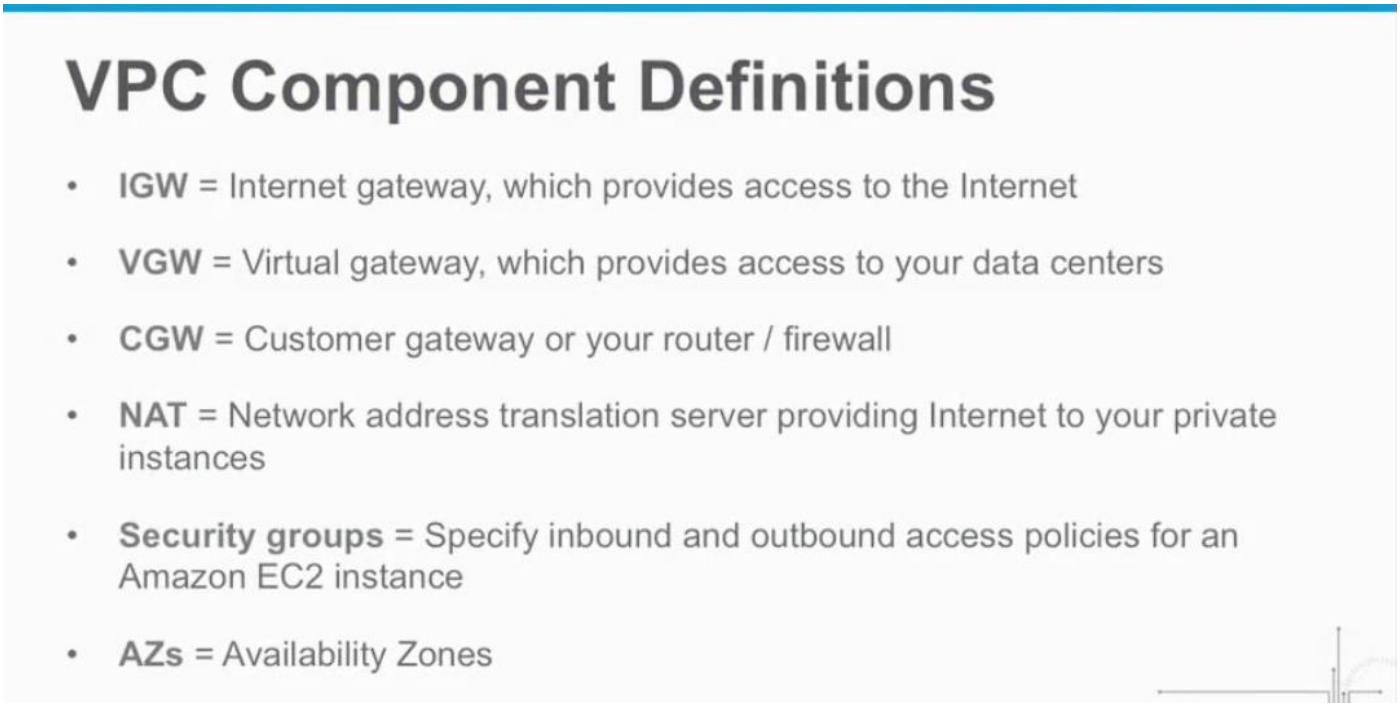
Here we have the same application running except we are not connecting to it through an IGW, but connecting to it through a VPN Gateway which exposes either DirectConnect or VPN access to this VPN to hit the private web apps inside it.



We are replicating our database server through DirectConnect or VPN to another server running in our VPC for Disaster Recovery, we will simply promote the VPC slave to a master and have our apps running again.



When you ***create a VPC***, you are ***choosing a CIDR block*** to use for your data center like a /16 block with lots of possible IP addresses. Then you ***create subnets within your VPC's larger CIDR block*** and ***deploy ACLs or NACLs*** (security at the network layer!) around particular subnet blocks to specify the ingress and egress policies allowed for each particular subnet. We can then also ***implement security at the EC2 layer using Security Groups attached to the EC2 instances***. We then ***create Route Tables that specify how a particular subnet is allowed to route,*** what gateways can it use for egress, etc. We can then ***create a VPN connection to connect or VPC back to our own data center networks***, or use DirectConnect.



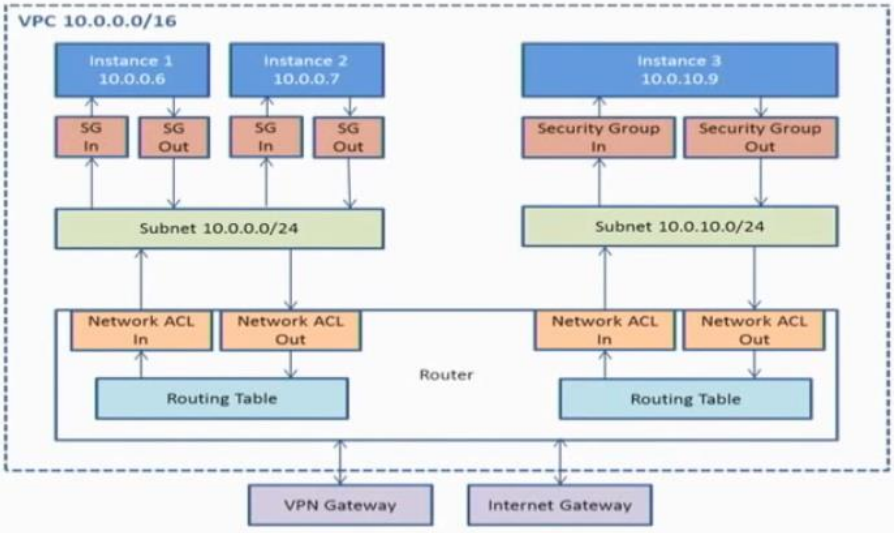
If you want to give access to your VPC, they need to come in through a VPN or through the internet. ***NAT*** is a service that you run inside your VPC to provide port address translation for services that are seeing private subnets that don't have internet access to get to the internet.

VPC Features

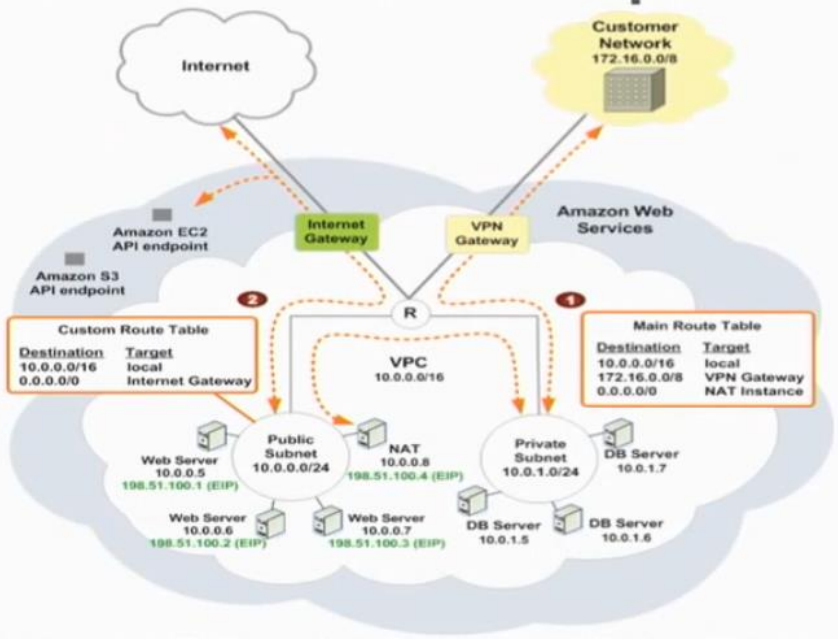
- Control of IP addressing CIDR block for your VPC
- Ability to subnet your VPC CIDR block
- Network access control lists
- Assign multiple IP addresses and multiple elastic network Interfaces
- Run private ELBs accessible from only within your VPC or over your VPN
- Bridge your VPC and your onsite IT infrastructure with private connectivity

A VPC provides you another set of IP address range that you can go and launch EC2 instances into.

Amazon VPC Network Security Controls



Virtual Private Cloud Example



Some VPC Considerations / Best Practices

- VPC CIDR block
- Subnets
- Network ACLs vs. security groups

Why Move to VPC?

All new accounts today already default to VPC* for EC2 and many other products. What does this mean?

What Is Default VPC / Default Subnet?

- Default VPC
  - Special VPC that is used with services when new accounts don't specify a target VPC
    - Amazon EC2, Amazon Relational Database Service (RDS), Elastic Load Balancing, Amazon Elastic MapReduce (EMR), AWS Elastic Beanstalk
  - One default VPC per region
  - Configurable the same as other VPCs; e.g., adding more subnets
- Default Subnets in Default VPC
  - Special subnet automatically created for each AZ for new accounts

Functionalities Delivered to EC2 by Move to VPC

- Static private IP address allocation
- Multiple IP address allocation and multiple ENIs
- Dynamic security group membership configuration
- Outbound packet filtering by security group
- Network access control lists (ACLs)
- Private ELBs

Connecting VPC with Your Data Centers

VPC Connectivity Options

- VPN connectivity

  Connect dual redundant tunnels between your on-premises equipment and AWS

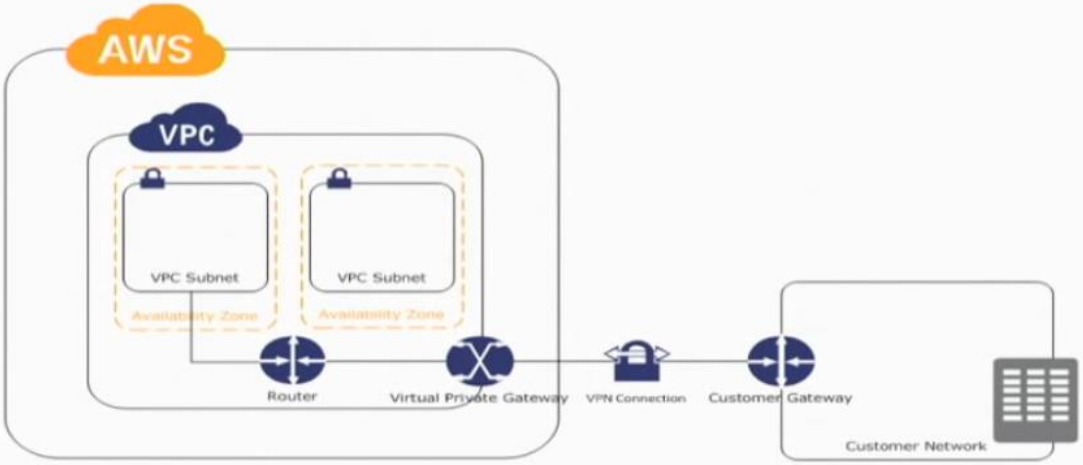
- AWS Direct Connect

  Establish a private network connection between your network and one of the AWS Regions

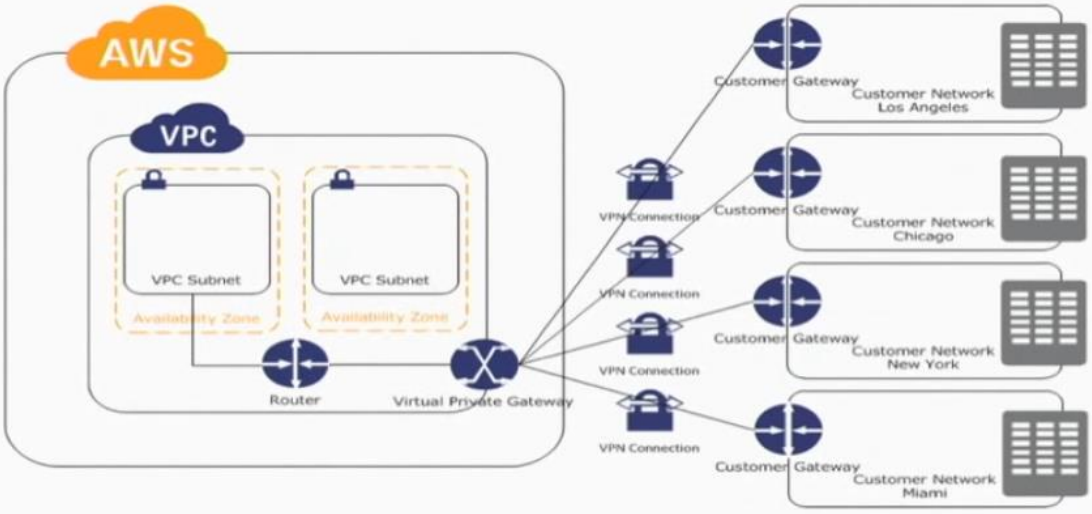
VPN Connectivity

- Redundant IPsec tunnels
- Supports BGP and static routing
- Redundant customer gateways

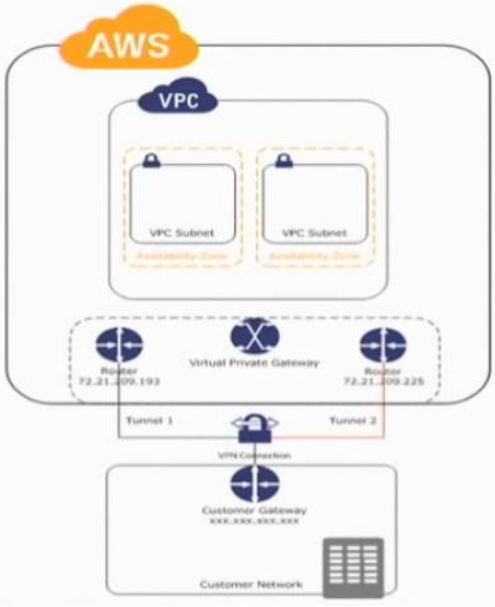
Single VPN Connection



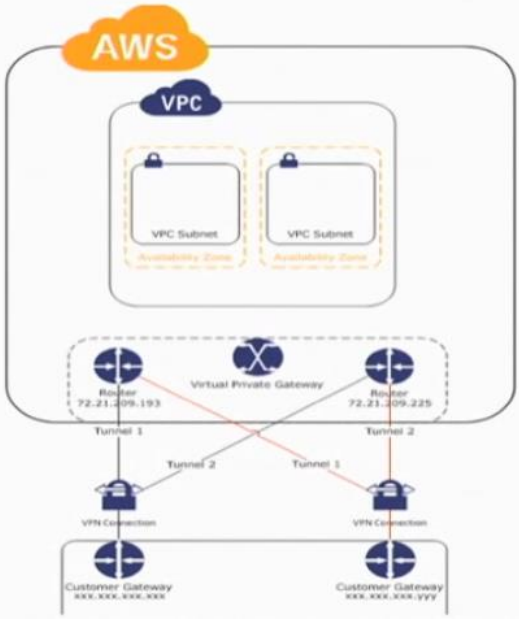
Multiple VPN Connections



Redundant Tunnels for Your VPN Connection



Redundant Customer Gateways



What is AWS Direct Connect?

- Alternative to using the Internet to access AWS cloud services
- Private network connection between AWS and your data center
- Can reduce costs, increase bandwidth, and provide a more consistent network experience than Internet-based connections

Why AWS Direct Connect?

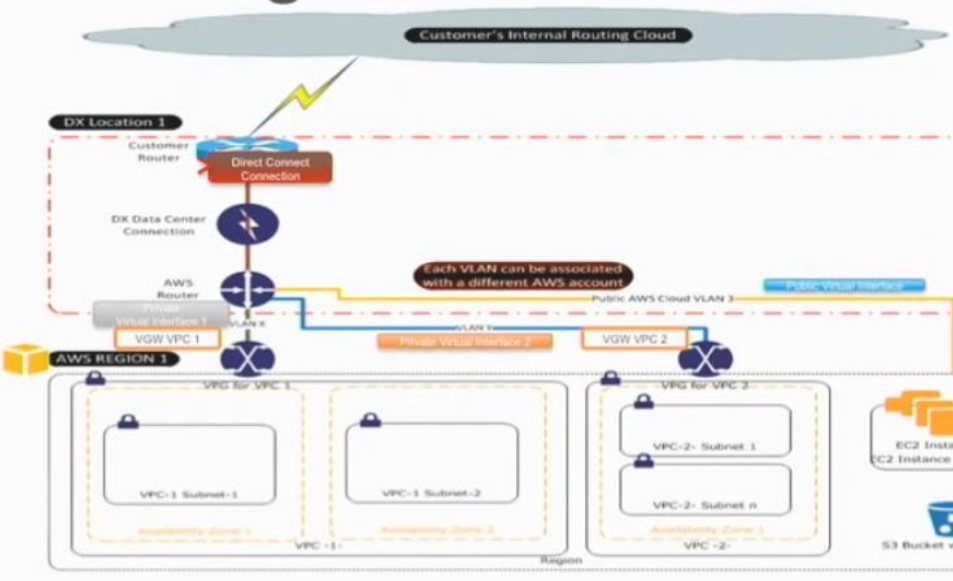
- Reduces your bandwidth costs
- Consistent network performance
- Compatible with all AWS services
- Private connectivity to your Amazon VPC

We have many AWS Direct Connect locations.

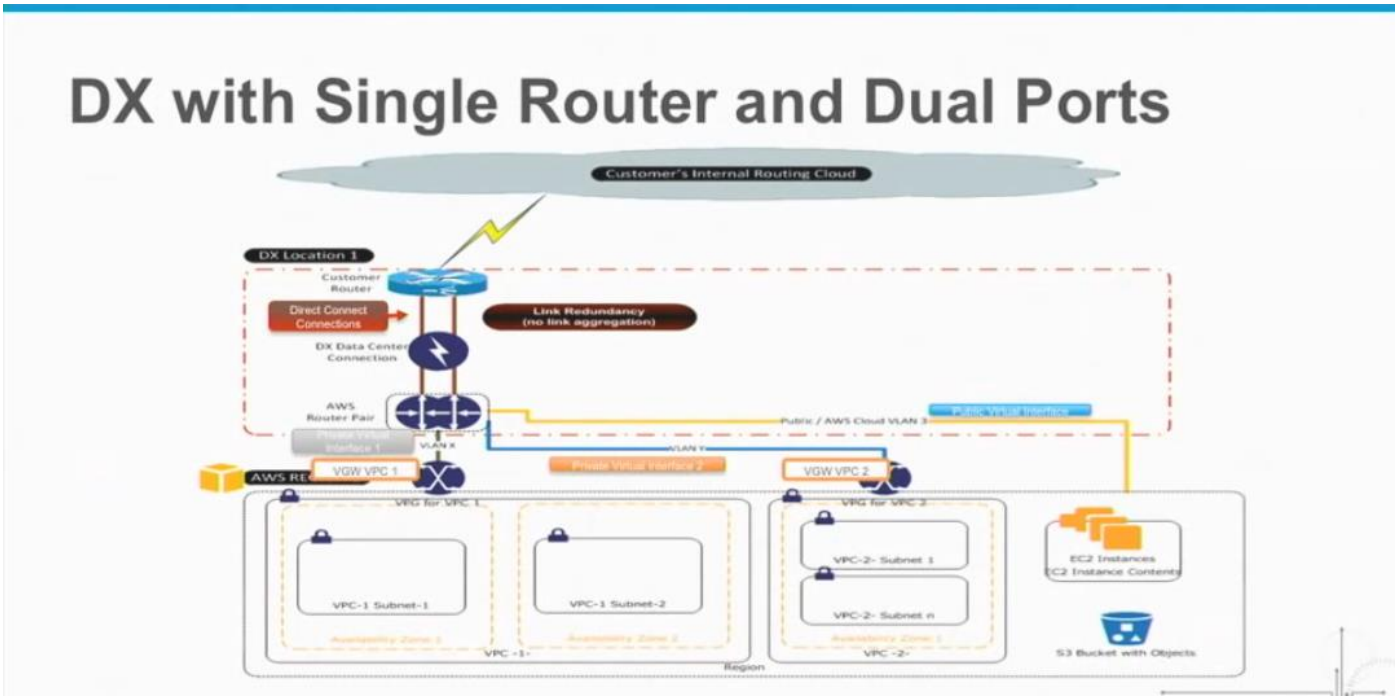
http://aws.amazon.com/directconnect/#details

Let's look at some Direct Connect architectures.

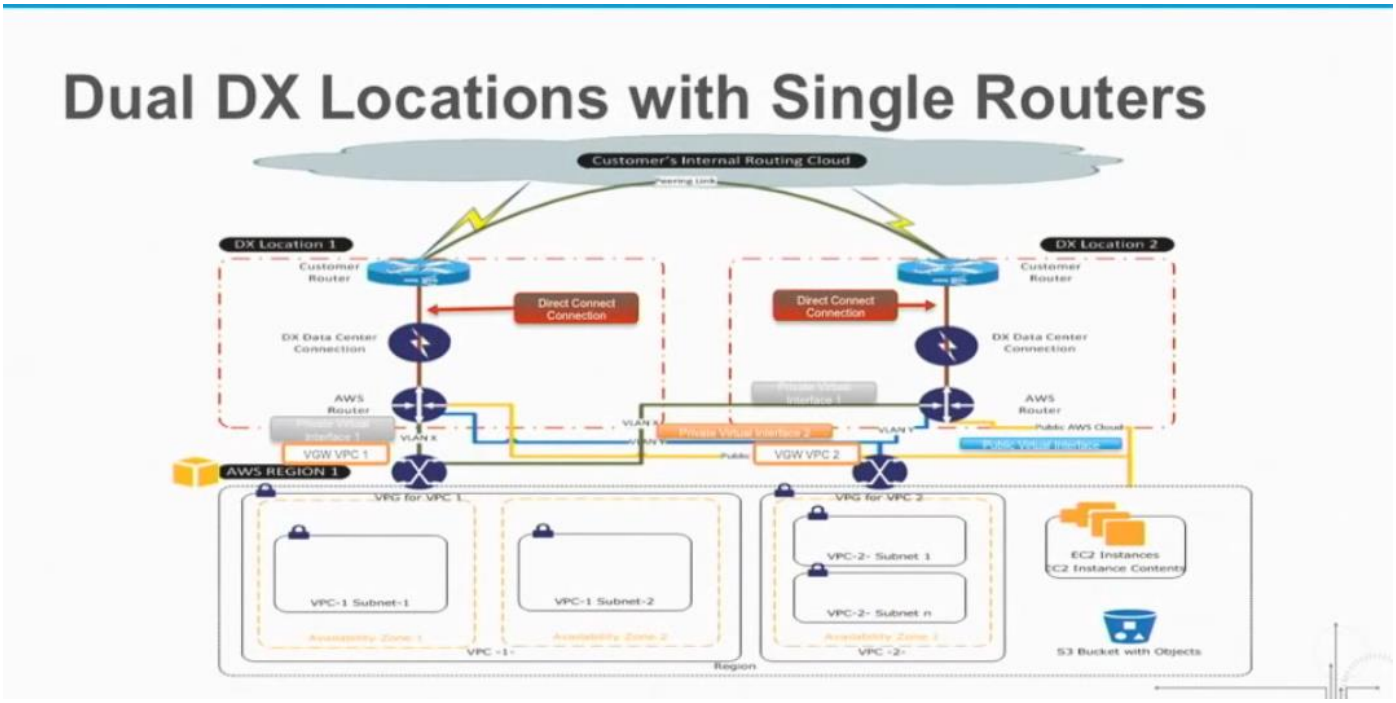
DX with Single Router Port



This is basic DirectConnect with no redundancy built in.



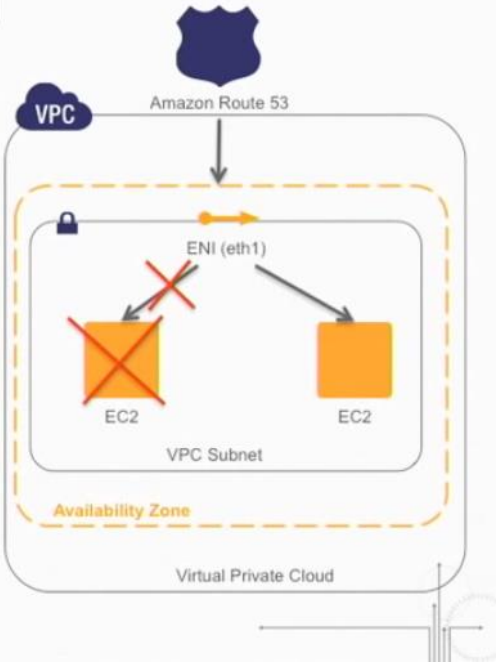
This is a more robust and recommended solution with dual routers being used.



Let's look at some design patterns for making your VPC infrastructure highly available.

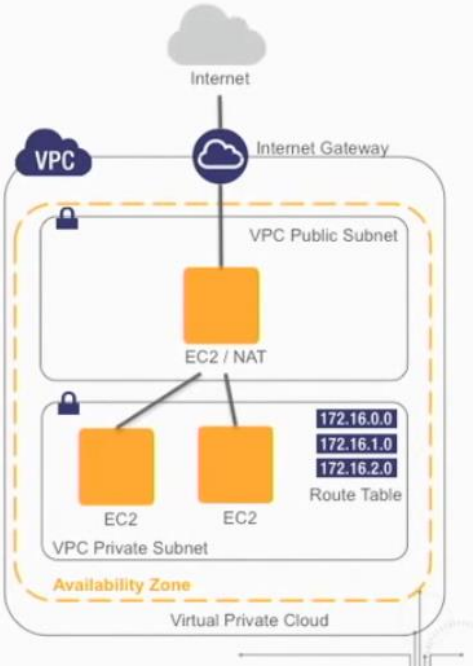
Floating Interface Pattern

- **Problem**
  If my instance fails or I need to upgrade it, I need to push traffic to another instance with the same public and private IP addresses and same network interface
- **Solution**
  Deploy your application in VPC and use an elastic network interface (ENI) on eth1 that can be moved between instances and retain same MAC, public, and private IP addresses
- **Pros**
  - Since we are moving the ENI, DNS will not need to be updated
  - Fallback is as easy as moving the ENI back to the original instance
  - Anything pointing to the public or private IP on the instance will not need to be updated.
  - ENIs can be moved across instances in a subnet



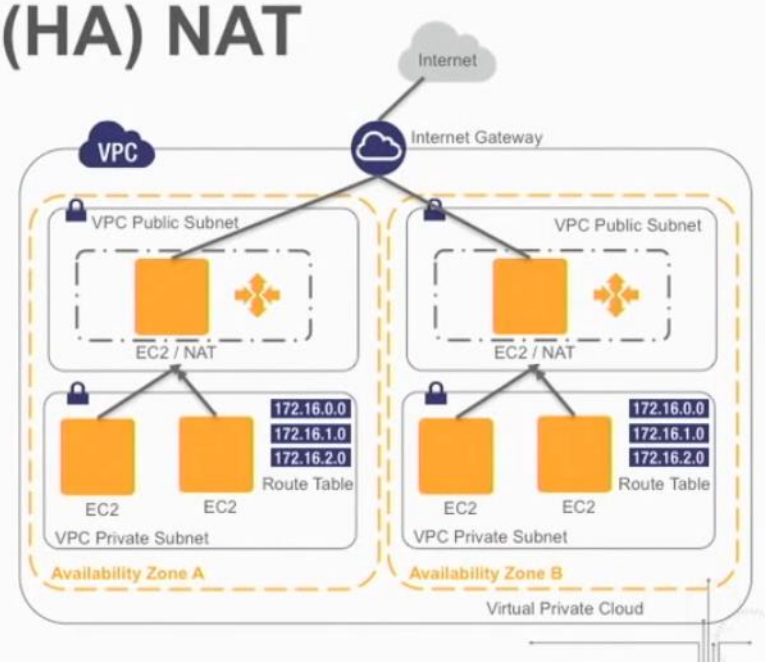
On Demand NAT in VPC

- **Problem**
  EC2 instances in a private subnet need access to the Internet to call APIs, for downloads and updates to software packages and the OS
- **Solution**
  Deploy a NAT server on an EC2 instance that will provide Internet access to servers in private subnets
- **Pros**
  - Your devices are not publicly addressable but still have Internet access
  - NAT gives instances in private subnet capability to access AWS services and APIs outside of VPC



High Availability (HA) NAT

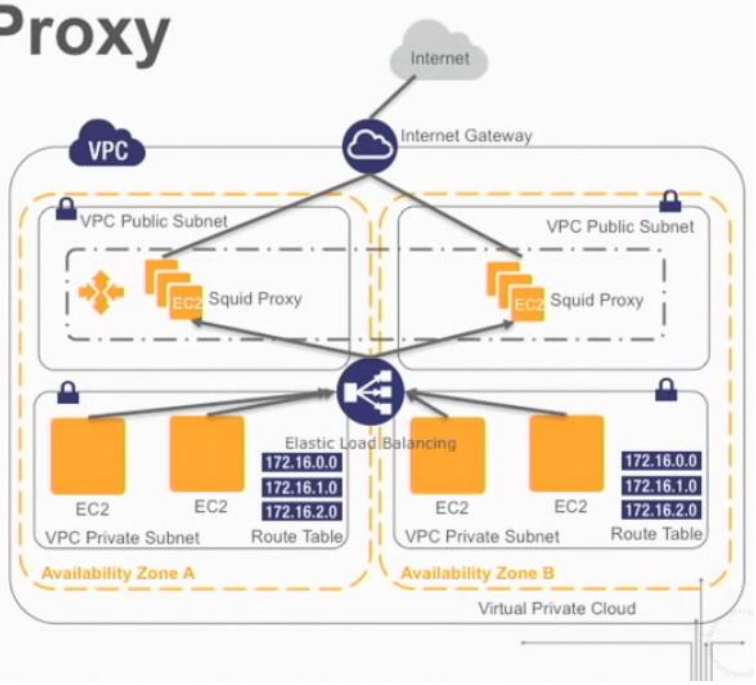
- **Problem**
  NAT inside of VPC is confined to a single instance, which could fail
- **Solution**
  - Run NAT in independent ASGs per AZ.
  - If NAT instance goes down, Auto Scaling will launch new NAT instance
  - As part of launch config, assign a public IP and call VPC APIs to update routes
- **Pros**
  - The NAT application is more HA with limited downtime



The ASG will be set to have a min of 1 and a max of 1 to make sure that that one instance is always healthy even if an instance dies.

HA NAT – Squid Proxy

- **Problem**
  - Standard NAT inside of VPC is confined to a single instance, which could fail
  - I also need to perform large puts and gets to Amazon S3
- **Solution**
  - Run Squid in proxy configuration in an ASG
  - On boot, configure instances to point to proxy for all HTTP(S) requests
- **Pros**
  - If a Squid proxy server dies, there are many and it will self heal and scale based on ASG policies
  - Much greater throughput can be achieved here as there is not a single-server per route table
- **Notes**
  - This is great for high-throughput requirements to get and put in Amazon S3 or elsewhere outside of the VPC
  - Need to manage a separate cluster of servers so this is more costly and requires more management



Next, let's look at some design patterns for making your application highly available.

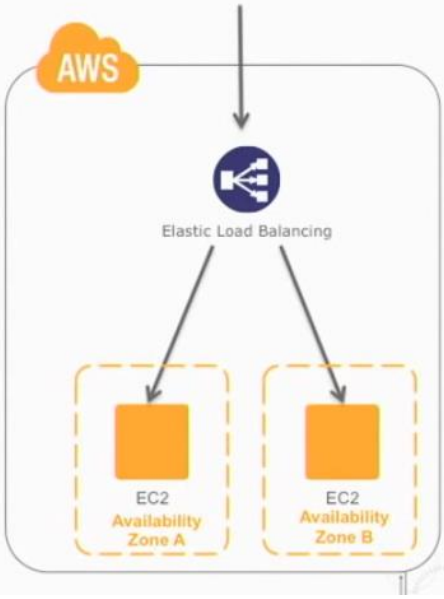
Availability	DPM	Downtime Per Year (24x365)		
99.000%	10000	3 Days	15 Hours	36 Minutes
99.500%	5000	1 Day	19 Hours	48 Minutes
99.900%	1000		8 Hours	46 Minutes
99.950%	500		4 Hours	23 Minutes
99.990%	100			53 Minutes
99.999%	10			5 Minutes
99.9999%	1			30 Seconds

"High Availability"

DMP—Defects per Million

Multi–Data Center Pattern

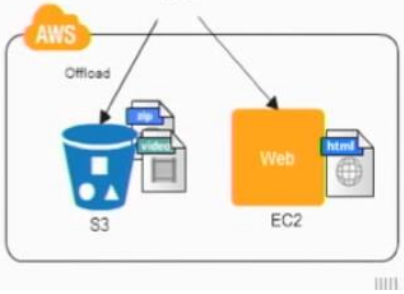
- **Problem**
  Increase availability of my application as everything fails when you least expect it
- **Solution**
  Distribute load between instances using Elastic Load Balancing across multiple AZs
- **Pros**
  - If an EC2 instance fails, the systems is still available as a whole
  - If an Availability Zone fails, the system is still available as a whole
  - Using Auto Scaling, you can add or replace with new instances when instances become unhealthy
- **Notes**
  - Need to store user-generated data in a common location such as Amazon S3 or NFS
  - Need to use sticky sessions or move session state off of web server



Web Storage Pattern

- **Problem**
  - Delivery of large files from a web server can become a problem in terms of network load
  - User generated content needs to be distributed across all my web servers
- **Solution**
  - Store static asset files in Amazon S3 and deliver the files directly from there
  - Objects that are stored in S3 can be accessed directly by users if set to being public
- **Pros**
  - The use of Amazon S3 eliminates the need to worry about network loads and data capacity on your web servers
  - Amazon S3 performs backups in at least three different data centers, and thus has extremely high durability.
  - The CloudFront CDN can be leveraged as a global caching layer in front of S3 to accelerate content to your end users

Yes, you can technically ship your static objects to AWS in a box with AWS Import / Export



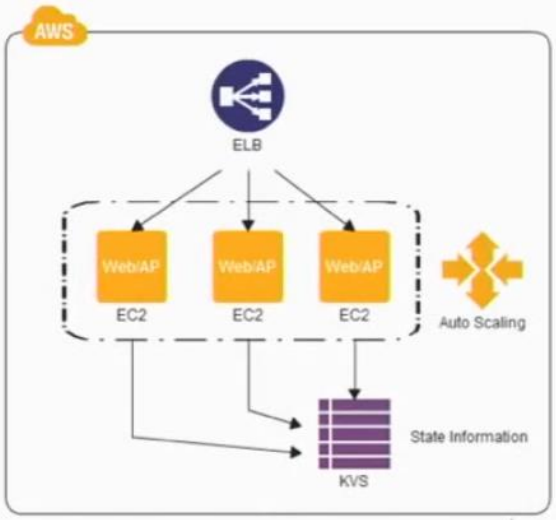
State Sharing

- **Problem**

  State is stored on my server so scaling horizontally does not work that well
- **Solution**
  - In order to scale horizontally and not have a user locked into a single server, I need to move state off of my server into a KVS
  - Moving session data into Amazon DynamoDB or Amazon ElastiCache allows my application to be stateless
- **Pros**

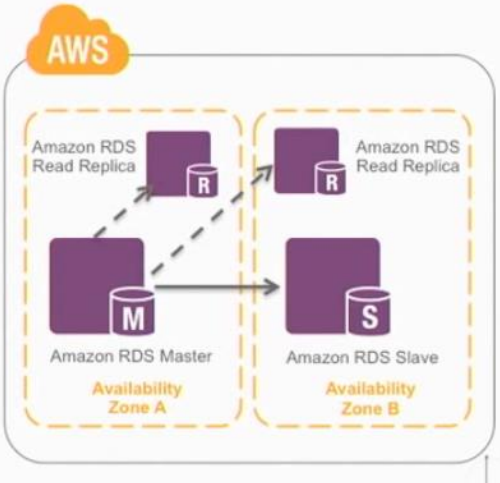
  This lets you use a scale-out pattern without having to worry about inheritance or loss of state information.
- **Notes**

  Because access to state information from multiple web/APP servers is concentrated on a single location, you must use caution to prevent the performance of the data store from becoming a bottleneck



High Availability Database Pattern

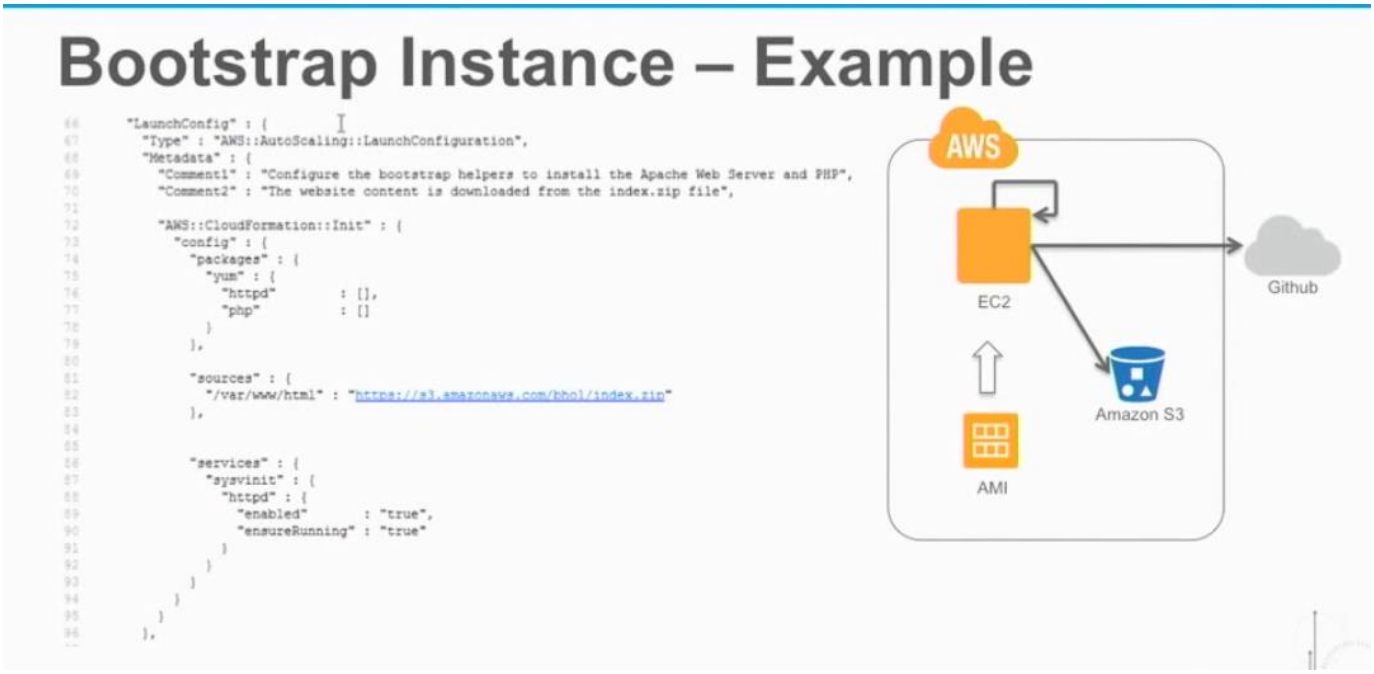
- **Problem**
  Need to have high availability solution that will withstand an outage of the DB master and can sustain high volume of reads
- **Solution**
  Deploy Amazon RDS with a master and slave configuration. In addition, deploy a read replica in each Availability Zone for reads and offline reporting
- **Pros**
  - One connection string for master and slave with automatic failover (takes approx. 3 min.) creates an HA database solution
  - Maintenance does not bring down DB but causes failover
  - Read replicas take load off of master so overall solution provides greater I/O for reads and writes



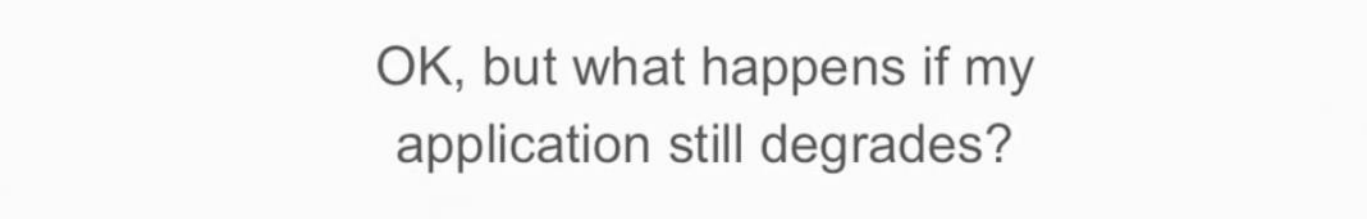
Use synchronous replication between the master DB and the slave DB. When the master goes down, you simply change the configuration file to point to the slave DB as the new master and move the master to slave when it gets back up.

Bootstrap Instance

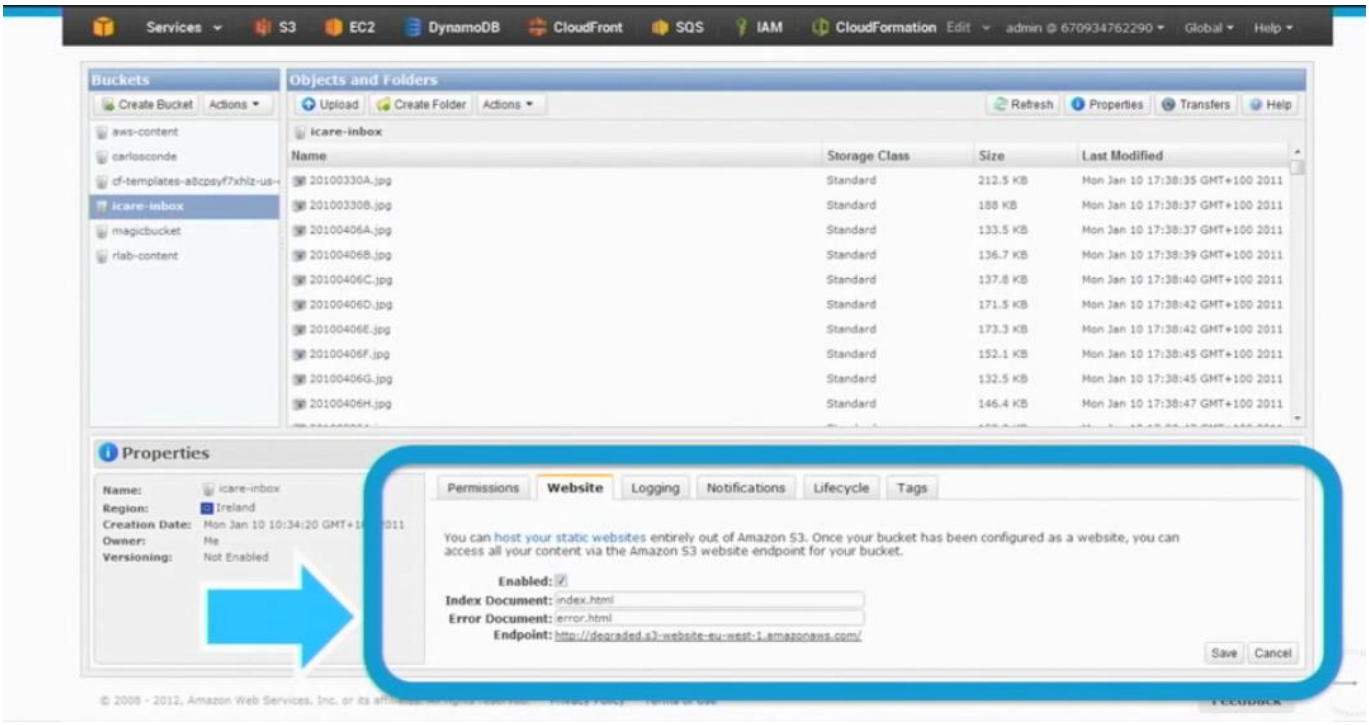
- **Problem**
  Code releases happen often and creating a new AMI every time you have a release and managing these AMIs across multiple regions adds complexity
- **Solution**
  Develop a base AMI, and then bootstrap the instance during the boot process to install software, get updates, and install source code so that your AMI rarely changes
- **Pros**
  Do not need to update AMI regularly and move customized AMI between regions for each software release
- **Notes**
  - During boot, it will most likely take more time to install and perform configuration than it would with a golden AMI
  - Bootstrapping can also be done through Auto Scaling and AWS CloudFormation

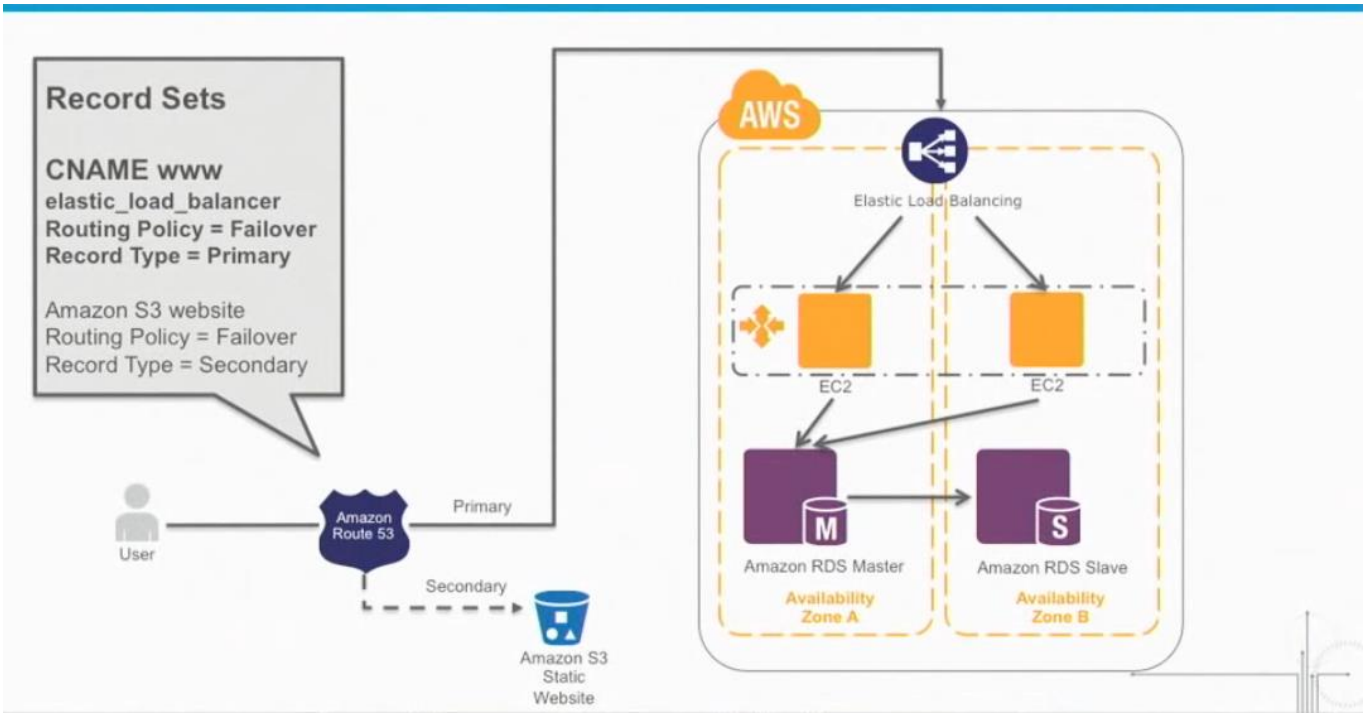


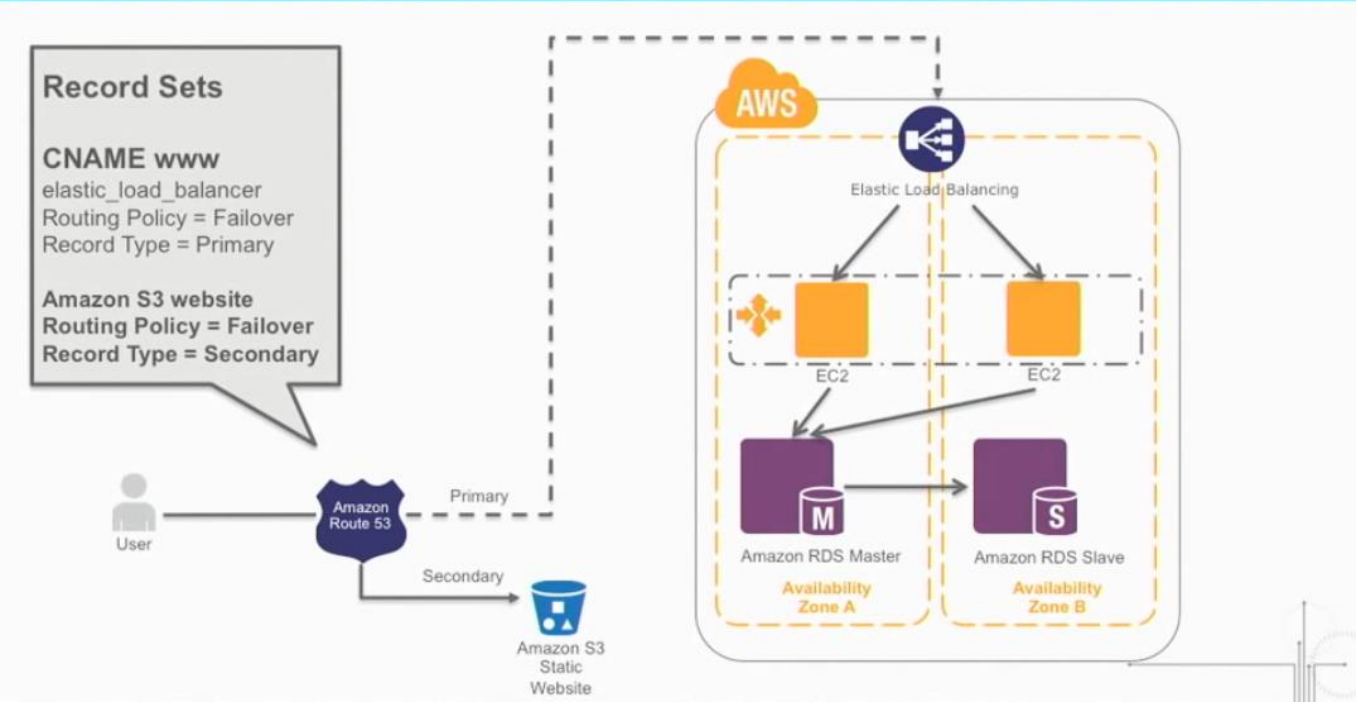
This is a CF template that does bootstrapping, we are using the CF ***init*** to bootstrap the application using an AMI base image during startup



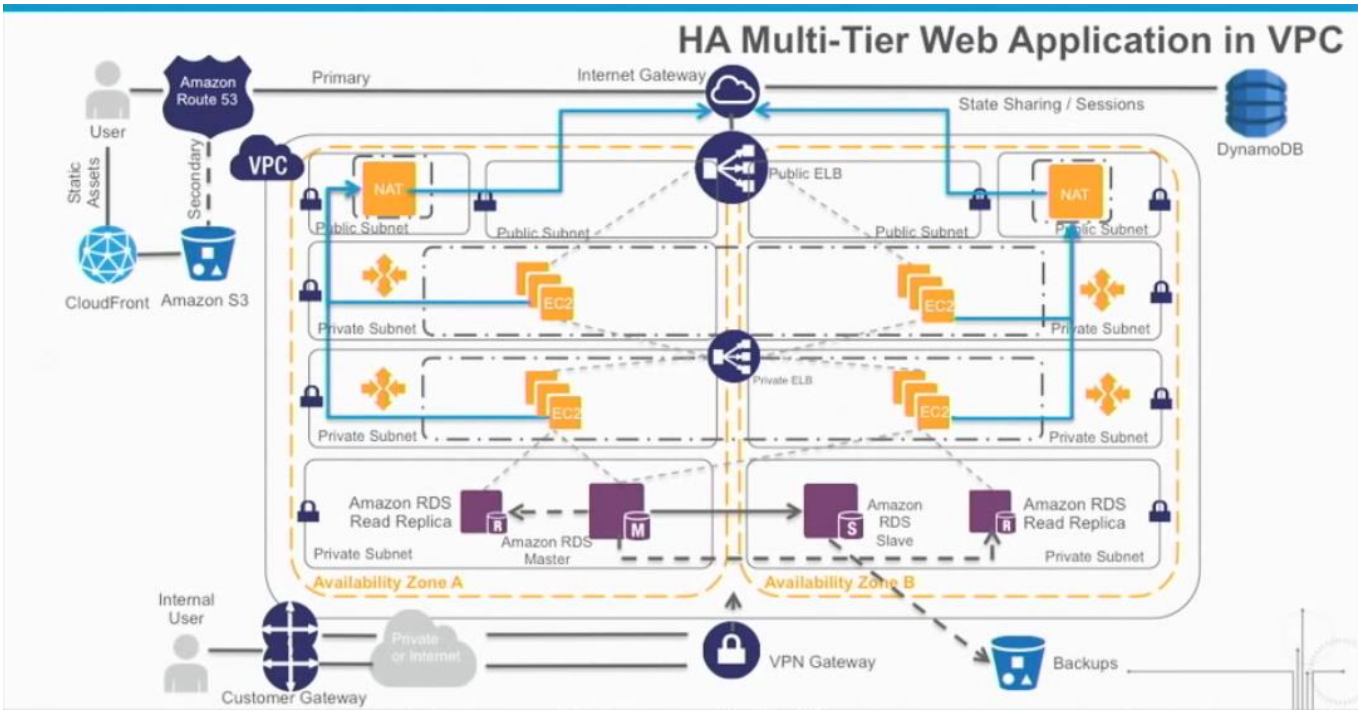
You can then use a DNS failover policy to switch to the static website version in S3 when your dynamic website running on your EC2 instances fail.


Services
S3
EC2
DynamoDB
CloudFront
SQS
IAM
CloudFormation
Buckets
Create Bucket
Actions
aws-content
carlosconde
icare-inbox
magicbucket
rlab-content
Objects and Folders
Upload
Create Folder
Refresh
Properties
Transfers
Help
Name
Storage Class
Size
Last Modified
Properties
Name:
Region:
Creation Date:
Owner:
Versioning:
Permissions
Website
Logging
Notifications
Lifecycle
Tags
You can host your static websites entirely out of Amazon S3. Once your bucket has been configured as a website, you can access all your content via the Amazon S3 website endpoint for your bucket.
Enabled:
Index Document: index.html
Error Document: error.html
Endpoint:
Save
Cancel


Record Sets
CNAME www
elastic_load_balancer
Routing Policy = Failover
Record Type = Primary
Amazon S3 website
Routing Policy = Failover
Record Type = Secondary
User
Amazon Route 53
Primary
Secondary
Amazon S3 Static Website
AWS
Elastic Load Balancing
EC2
EC2
M
S
Amazon RDS Master
Amazon RDS Slave
Availability Zone A
Availability Zone B



So what might a highly available application VPC look like using the best practices we learned?



Testing Our Highly Available Application

You can use Bees Up to spin up a set of EC2 instance bees that will then run request load on your apps for testing.

Load and Fault Testing Tools

- Apache Bench
- Bees with Machine Guns
- HP LoadRunner
- Chaos Monkey

Chaos Monkey

- What is Chaos Monkey?
  - Chaos Monkey targets and terminates instances in a region
  - Implementations
    - Open source Java code for a service implementation
    - Command-line tool
- Why run Chaos Monkey?
  - Failures happen when you least expect it
  - Best to be prepared by testing
- Auto Scaling groups
  - Targets terminating instances in Auto Scaling groups
- Configuration
  - Opt in or out model
  - Tunable so you can terminate one instance per ASG per day
  - At Netflix, Chaos Monkey runs Monday – Thursday 9AM – 3PM for random instance kill

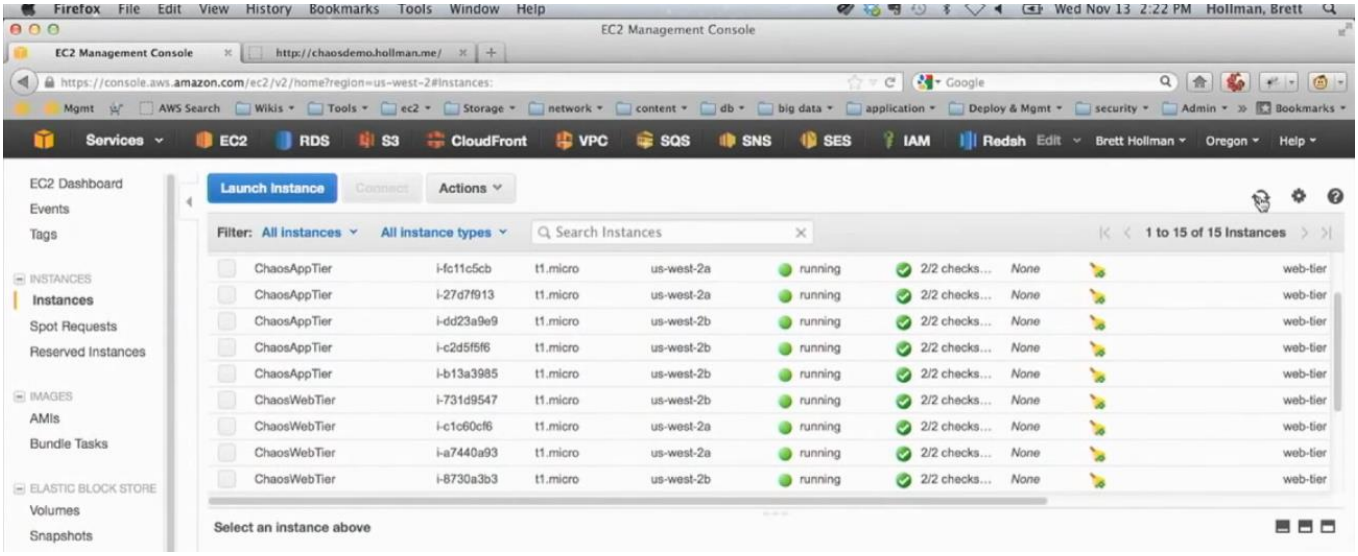
Chaos Monkey Demo

- We will demo Chaos Monkey against a mock three-tier application that has Auto Scaling groups at each layer
  - http://chaosdemo.hollman.me/

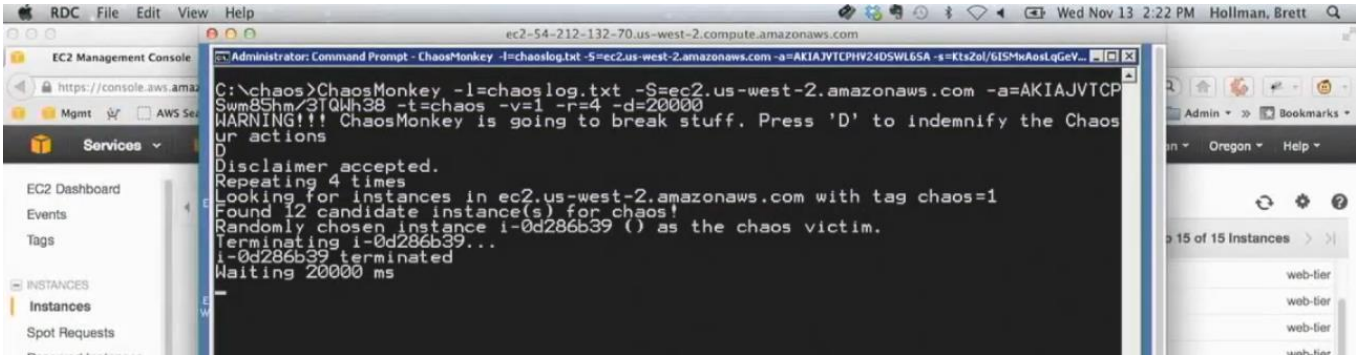
- Using Chaos Monkey CLI tool for demo

```
> ChaosMonkey
      -l=chaoslog.txt
      -S=ec2.us-west-2.amazonaws.com
      -a=XXXXXXXXXXXXXXXXXXXXXXXXX
      -s=XXXXXXXXXXXXXXXXXXXXXXXXXXXX
      -t=chaos
      -v=1
      -r=4
      -d=15000
```

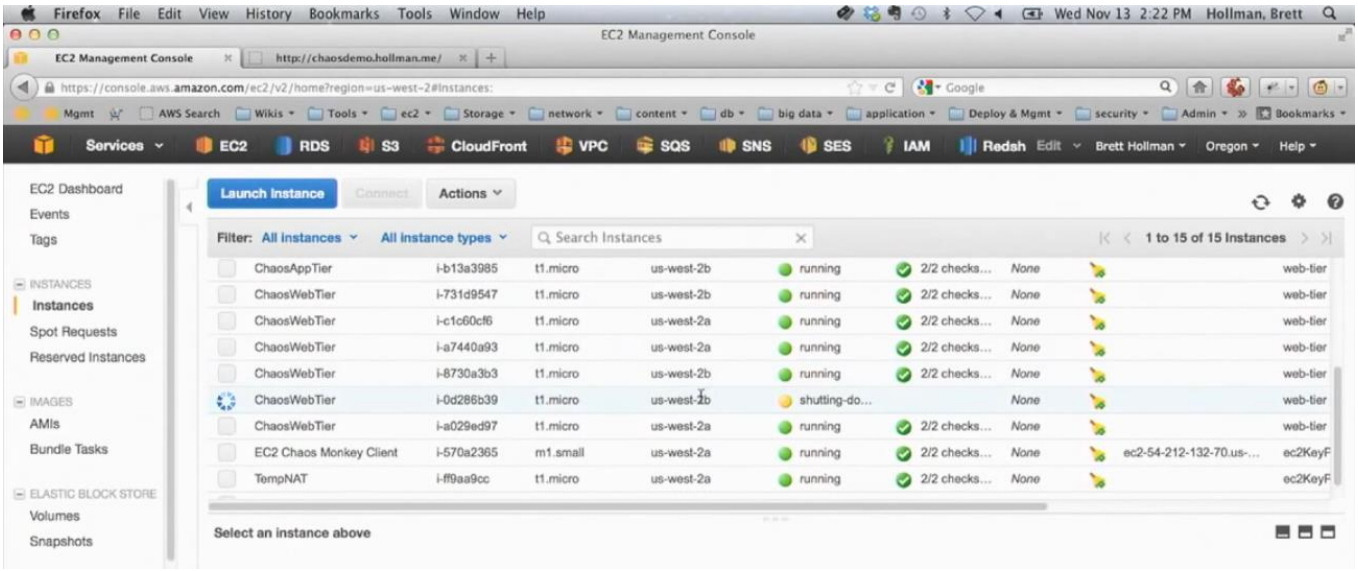
Chaos Monkey Demo



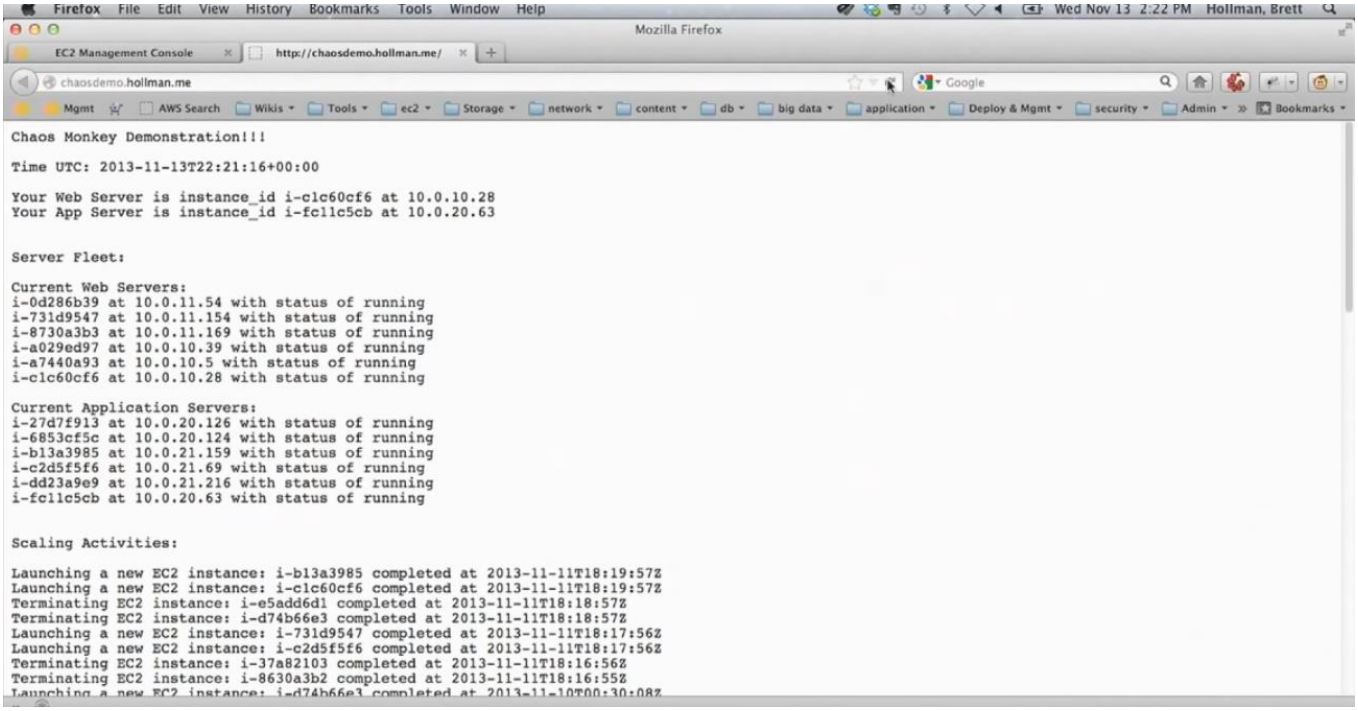
We have several EC2 instances running in our web and app tiers

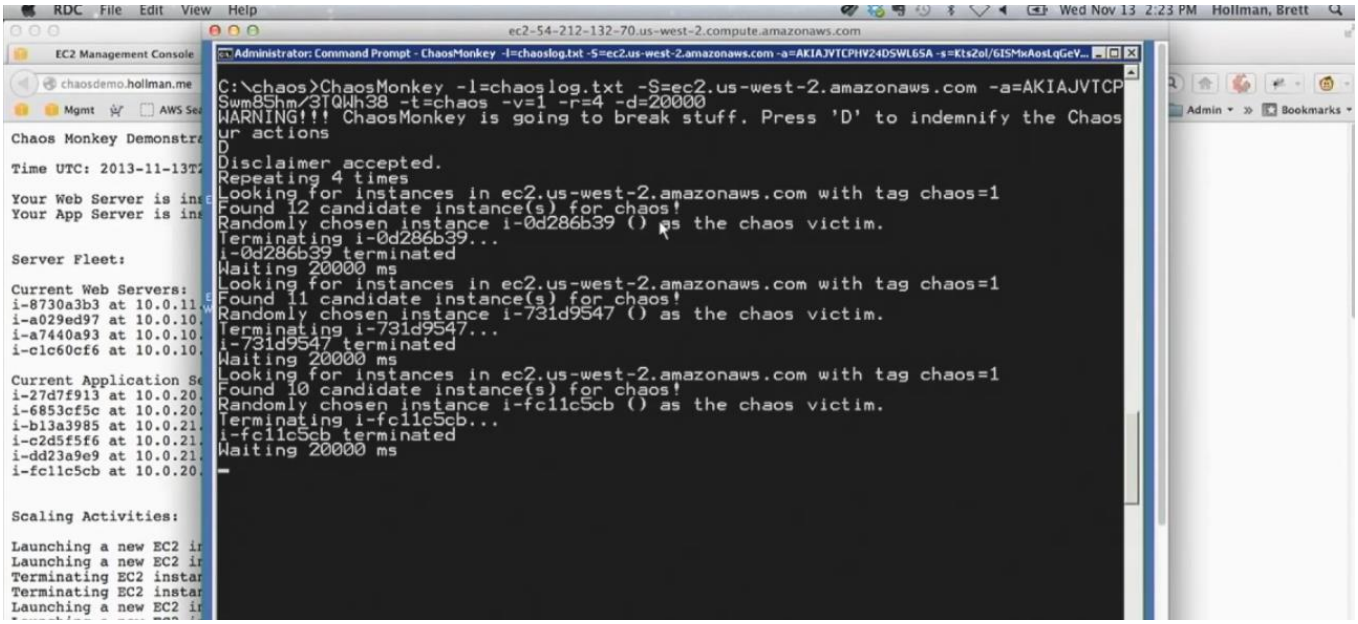


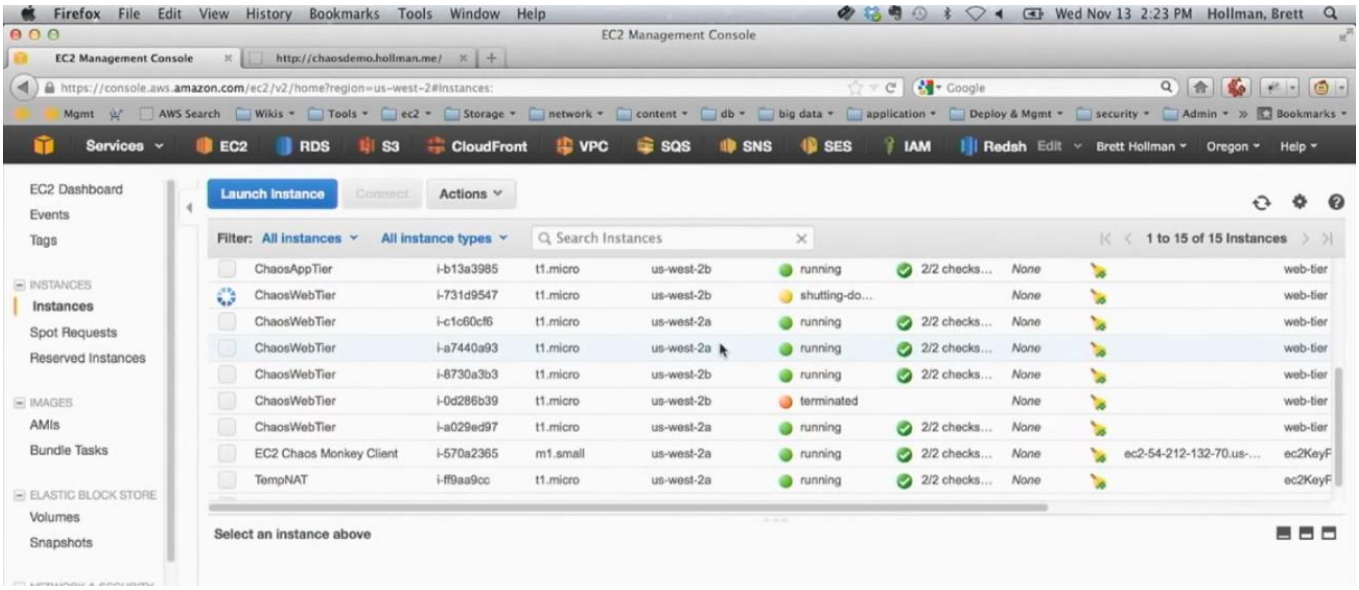
We use chaos monkey to start shutting down instances for resilience testing

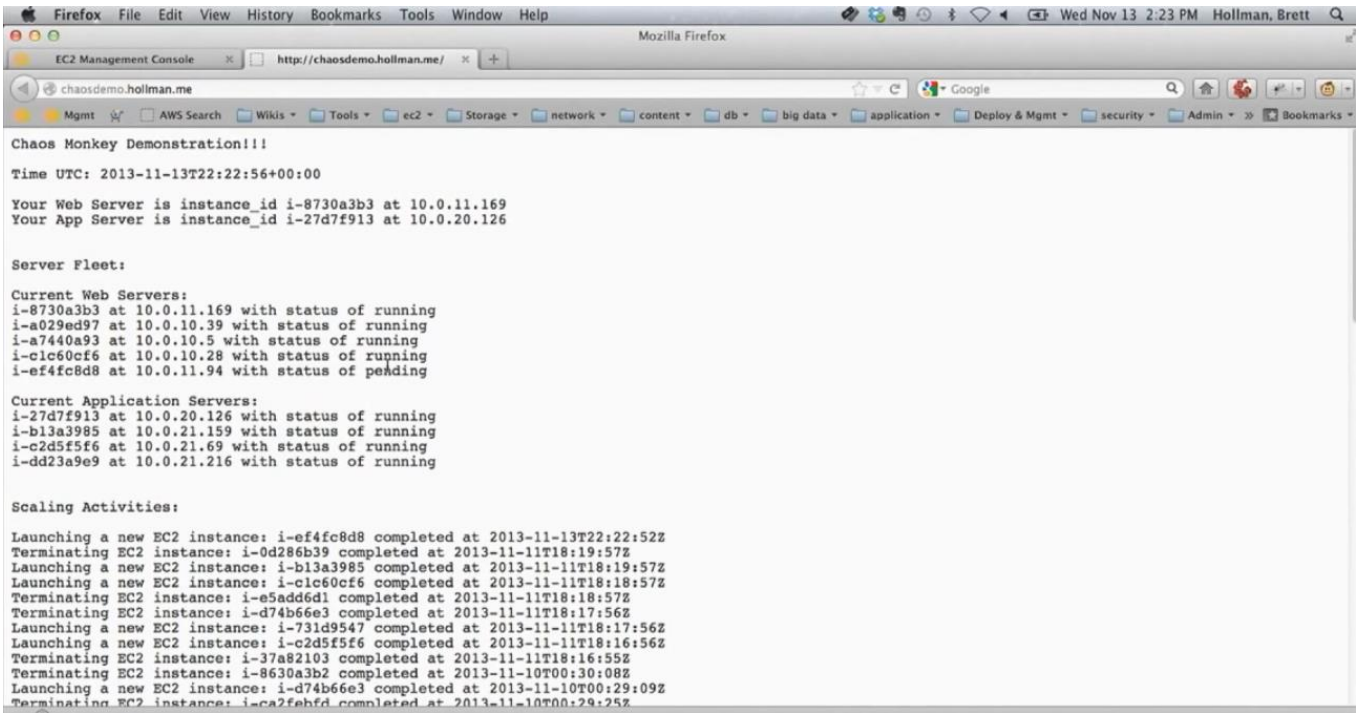


We can now see an instance shutting down in our web tier due to chaos monkey killing that instance

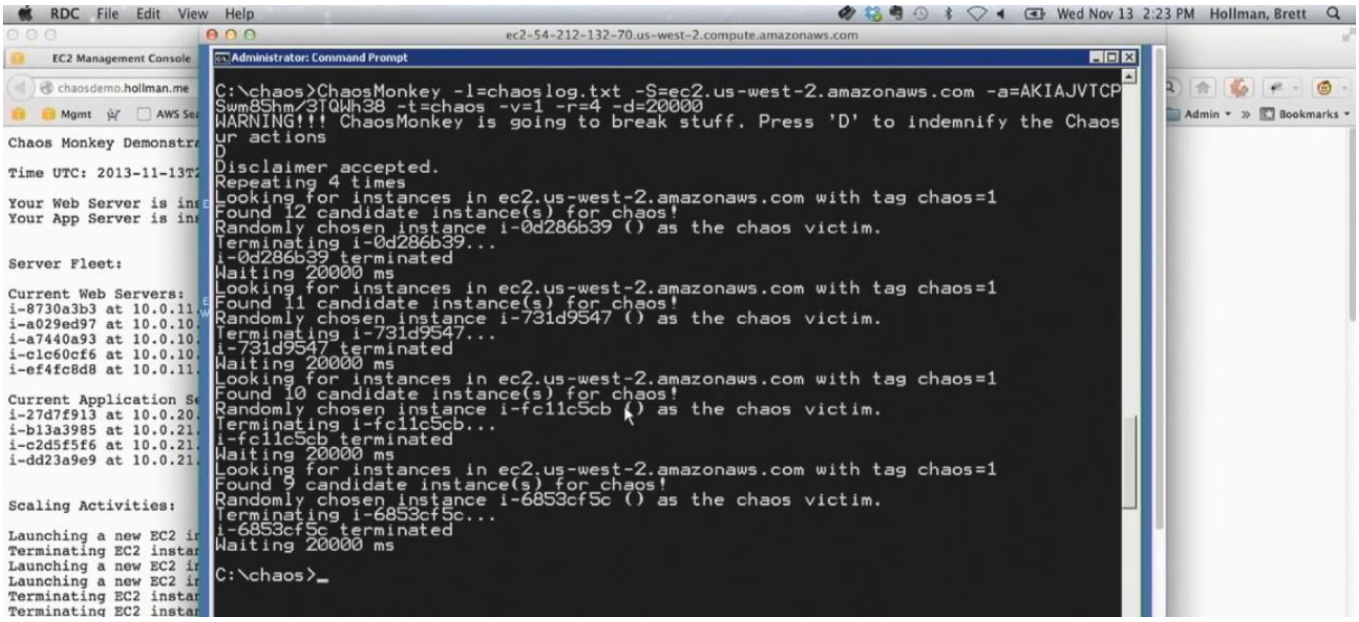

Firefox File Edit View History Bookmarks Tools Window Help
Wed Nov 13 2:22 PM Hollman, Brett
Mozilla Firefox
EC2 Management Console
http://chaosdemo.hollman.me/
chaosdemo.hollman.me
Chaos Monkey Demonstration!!!
Time UTC: 2013-11-13T22:21:16+00:00
Your Web Server is instance_id i-c1c60cf6 at 10.0.10.28
Your App Server is instance_id i-fc11c5cb at 10.0.20.63
Server Fleet:
Current Web Servers:
i-0d286b39 at 10.0.11.54 with status of running
i-731d9547 at 10.0.11.154 with status of running
i-8730a3b3 at 10.0.11.169 with status of running
i-a029ed97 at 10.0.10.39 with status of running
i-a7440a93 at 10.0.10.5 with status of running
i-c1c60cf6 at 10.0.10.28 with status of running
Current Application Servers:
i-27d7f913 at 10.0.20.126 with status of running
i-6853cf5c at 10.0.20.124 with status of running
i-b13a3985 at 10.0.21.159 with status of running
i-c2d5f5f6 at 10.0.21.69 with status of running
i-dd23a9e9 at 10.0.21.216 with status of running
i-fc11c5cb at 10.0.20.63 with status of running
Scaling Activities:
Launching a new EC2 instance: i-b13a3985 completed at 2013-11-11T18:19:57Z
Launching a new EC2 instance: i-c1c60cf6 completed at 2013-11-11T18:19:57Z
Terminating EC2 instance: i-e5add6d1 completed at 2013-11-11T18:18:57Z
Terminating EC2 instance: i-d74b66e3 completed at 2013-11-11T18:18:57Z
Launching a new EC2 instance: i-731d9547 completed at 2013-11-11T18:17:56Z
Launching a new EC2 instance: i-c2d5f5f6 completed at 2013-11-11T18:17:56Z
Terminating EC2 instance: i-37a82103 completed at 2013-11-11T18:16:56Z
Terminating EC2 instance: i-8630a3b2 completed at 2013-11-11T18:16:55Z


RDC File Edit View Help
Wed Nov 13 2:23 PM Hollman, Brett
ec2-54-212-132-70.us-west-2.compute.amazonaws.com
Administrator: Command Prompt - ChaosMonkey -l=chaoslog.txt -S=ec2.us-west-2.amazonaws.com -a=AKIAJVTCPHV24DSWL6SA -s=KtsZol/6ISMxAosLqGeV...
C:\chaos>ChaosMonkey -l=chaoslog.txt -S=ec2.us-west-2.amazonaws.com -a=AKIAJVTCP
Swm85hm/3TQWh38 -t=chaos -v=1 -r=4 -d=20000
WARNING!!! ChaosMonkey is going to break stuff. Press 'D' to indemnify the Chaos
ur actions
D
Disclaimer accepted.
Repeating 4 times
Looking for instances in ec2.us-west-2.amazonaws.com with tag chaos=1
Found 12 candidate instance(s) for chaos!
Randomly chosen instance i-0d286b39 () as the chaos victim.
Terminating i-0d286b39...
i-0d286b39 terminated
Waiting 20000 ms
Looking for instances in ec2.us-west-2.amazonaws.com with tag chaos=1
Found 11 candidate instance(s) for chaos!
Randomly chosen instance i-731d9547 () as the chaos victim.
Terminating i-731d9547...
i-731d9547 terminated
Waiting 20000 ms
Looking for instances in ec2.us-west-2.amazonaws.com with tag chaos=1
Found 10 candidate instance(s) for chaos!
Randomly chosen instance i-fc11c5cb () as the chaos victim.
Terminating i-fc11c5cb...
i-fc11c5cb terminated
Waiting 20000 ms

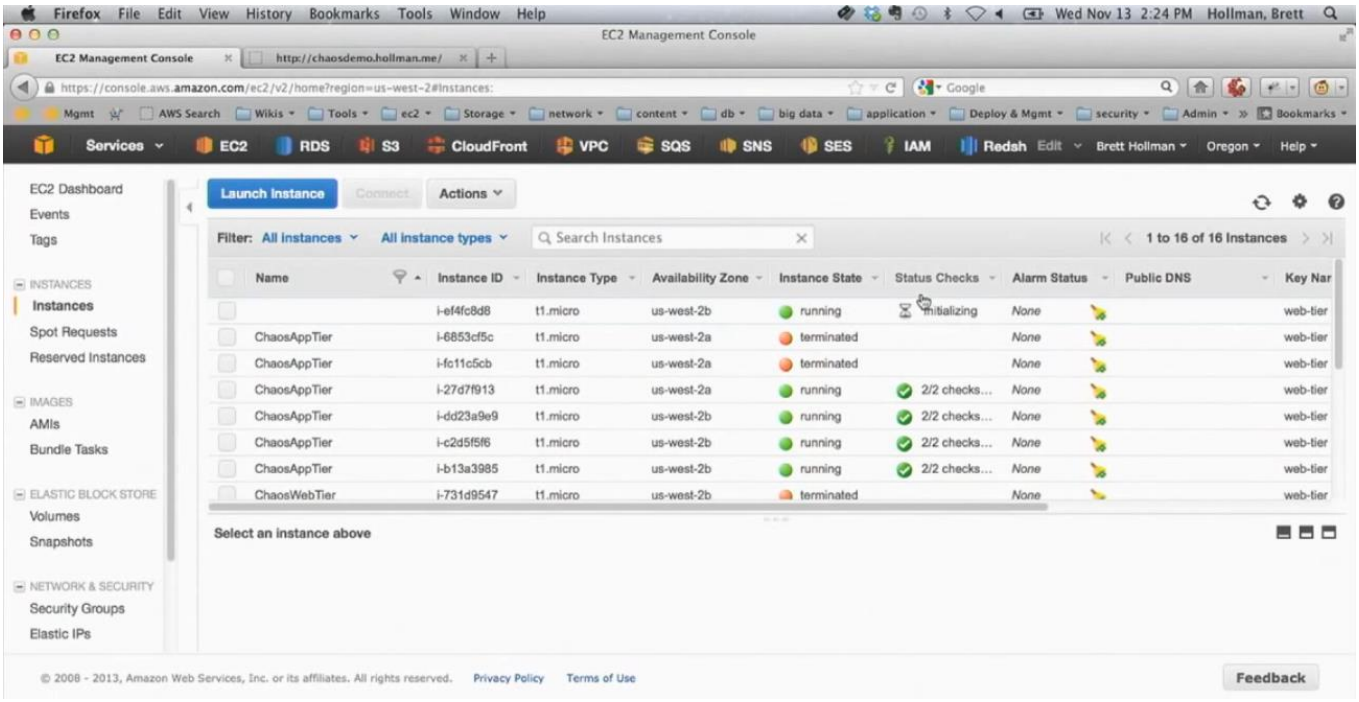


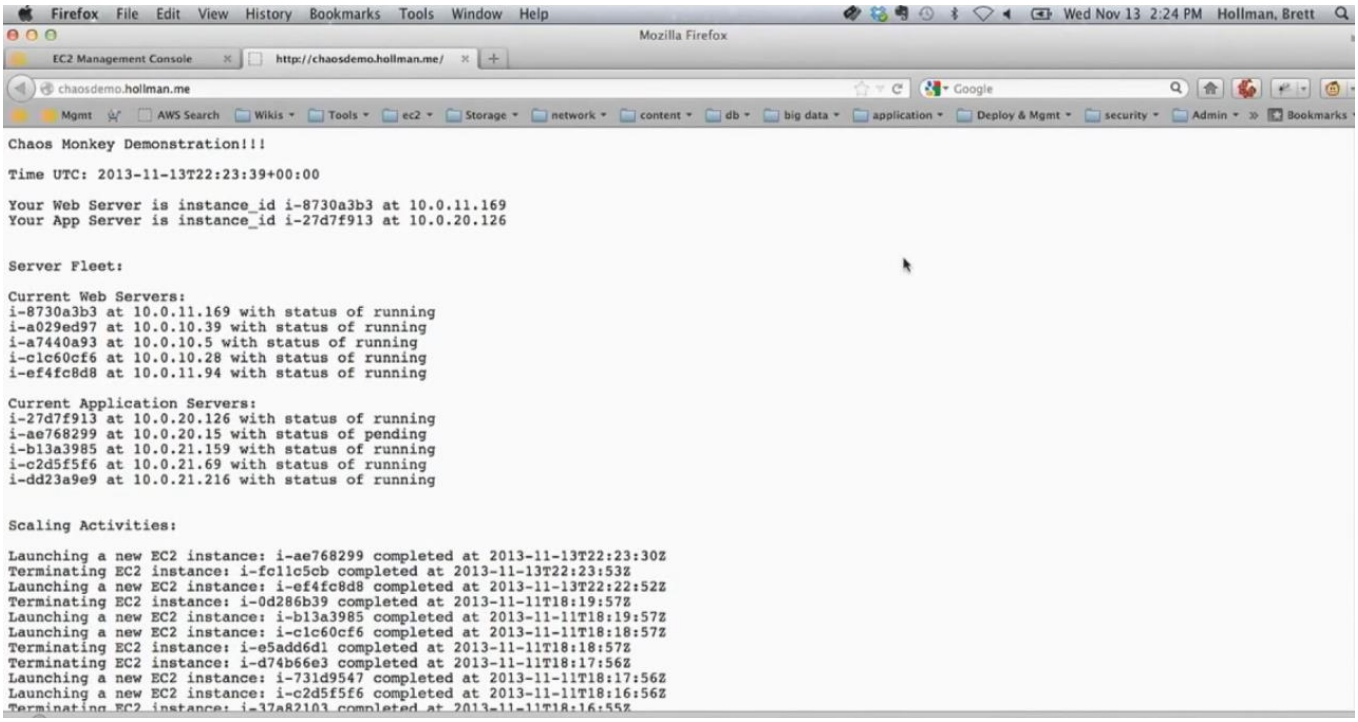


Our application is still working properly by trying to replace down servers

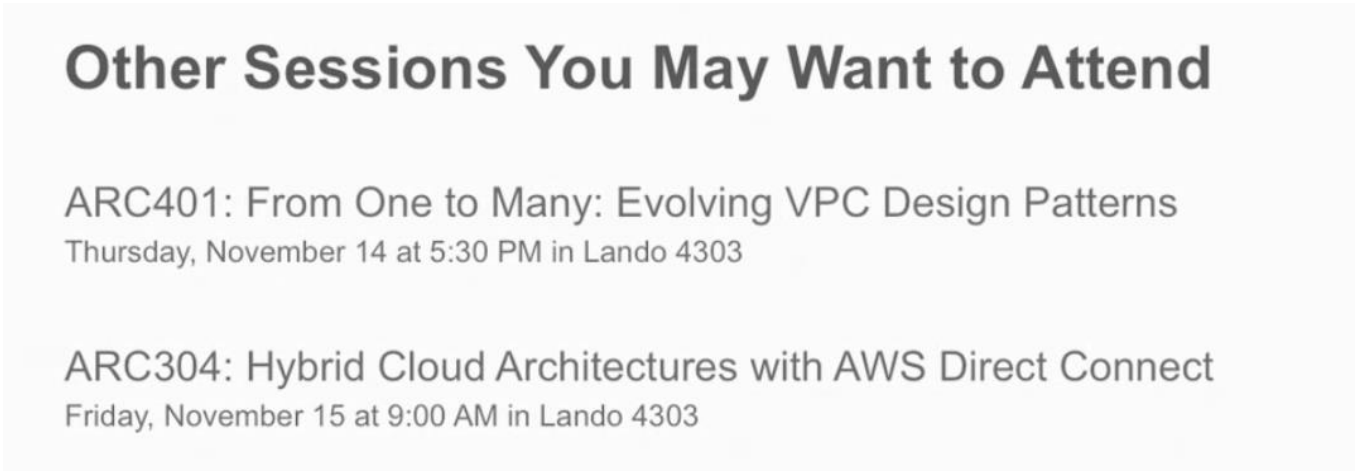


Chaos monkey is now shut down for the test





Our application is still working fine by rebuilding itself automatically



Startup Spotlight Sessions with Dr. Werner Vogels

Thurs. Nov 14, Marcello Room 4406

SPOT 203 - Fireside Chats – Startup Founders, 1:30-2:30pm

- Eliot Horowitz, CTO of MongoDB
- Jeff Lawson, CEO of Twilio
- Valentino Volonghi, Chief Architect of AdRoll

SPOT 204 - Fireside Chats – Startup Influencers, 3:00-4:00pm

- Albert Wegner, Managing Partner at Union Square Ventures
- David Cohen, Founder and CEO of TechStars

SPOT 101 - Startup Launches, 4:15-5:15pm

- 5 companies powered by AWS launching at AWS re:Invent 2013