Solutions Architect Hiring Assignment

July 2nd 2019

Hae-Yeon, Hwang

Executive Summary

My Goal

- Proposal AWS cloud architecture for early stage start-up company who trying to launch a web application on AWS

Consideration

- Proof of concept (PoC) stage, but want a deploy on AWS cloud.
- Face a too many problem (failed in loading page at LAMP environment).
- Guide a proper architecture to customer who need a clearly solution.

Assignment

Task #1

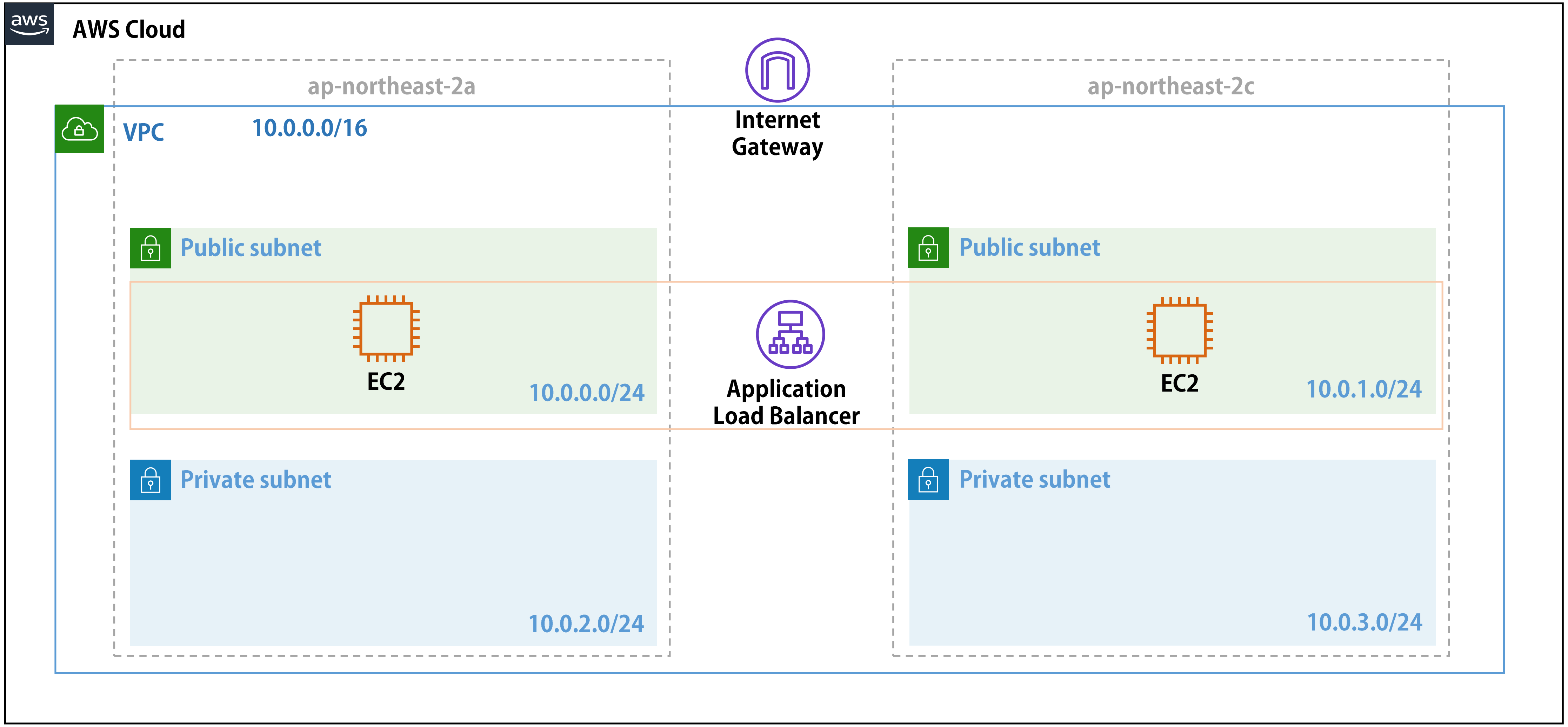
- Trouble shoot the issue by modifying the misconfiguration generated by given CloudFormation template and load demo.html page thru ELB hostname

Task #2

- Propose changes to the current implementation that would improve the reliability, security, cost, operation and performance before the project goes into production.

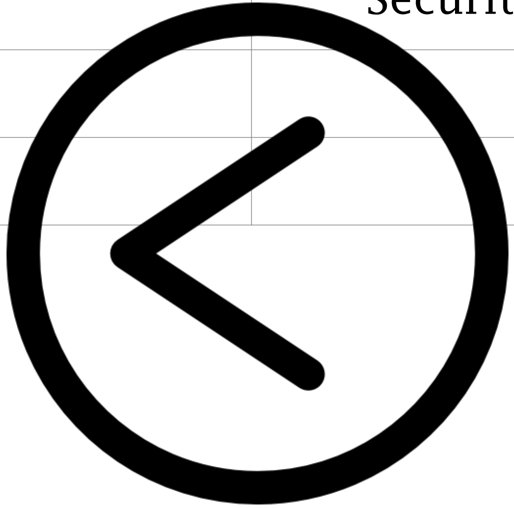
Task#1

Task #1 :: Target Architecture



Task #1 :: Comparison, On-premise vs. AWS

Comparison	On-Premise	AWS
Gateway	Router	Internet Gateway
Load Balancer	L4/L7 switch	ELB (classic/network/application)
Subnet	VLAN(802.1q), L2/L3 switch	Subnet
Filter	Ingress/egress Filter	Security Group (In/outbound Rule)
LB Instance	Physical Server [Farm]	ELB registered EC2 instance
Security Entity	Firewall / iptables	ACL



Easy to use

Flexible

Cost-Effective

Reliable

Secure

Scalable and high performance

Task #1 :: Trouble Shooting (step-by-step)

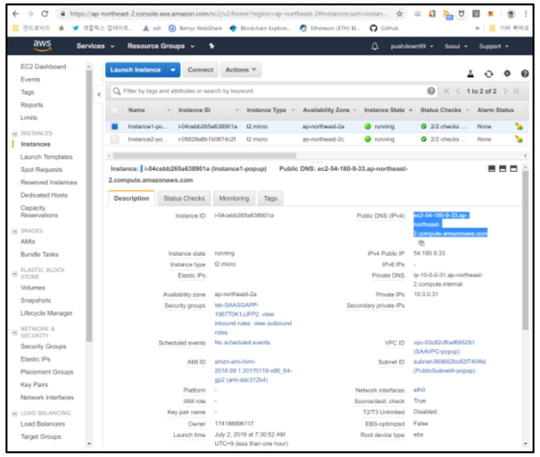
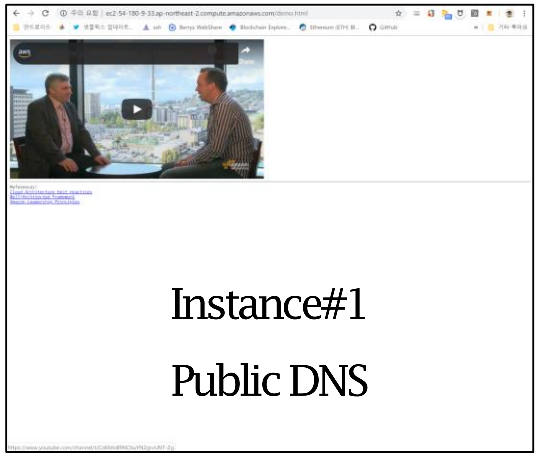
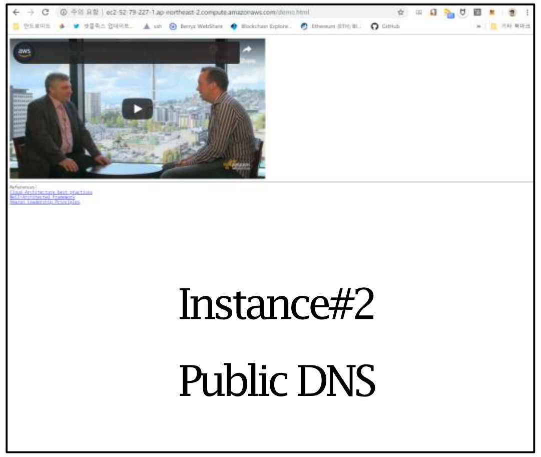
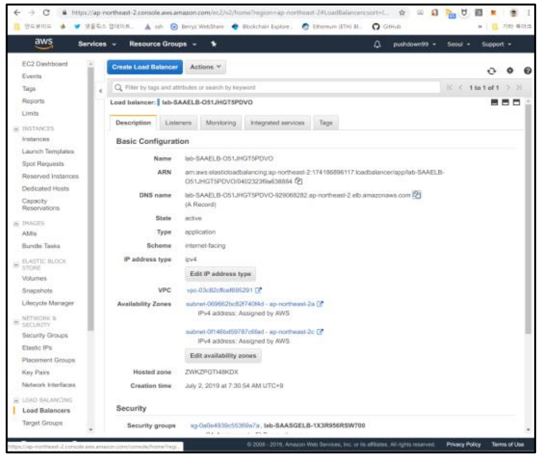
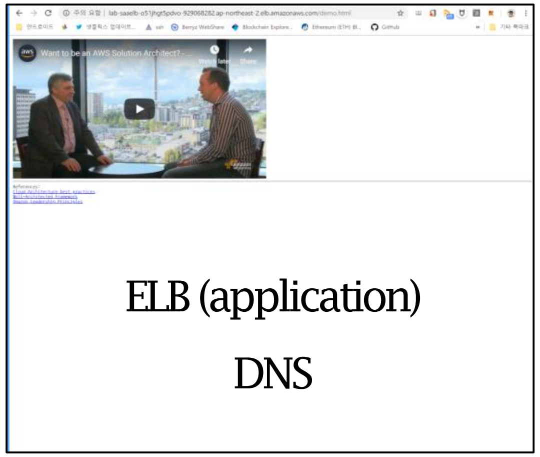
Build VPC

- VPC (My VPC, 10.0.0.0/16)
- Subnet (Public 1/10.0.1.0/24, Public 2/10.0.2.0/24; auto-assign IP setting)
- Internet Gateway (My IG; attach to VPC)
- Route Table (add route 0.0.0.0/0; subnet associations)
- Security Group (VPC; Inbound Rules: HTTP/Anywhere)

Launch a EC2 and ELB/ALB

- Launch EC2 (Amazon Linux AMI; VPC; Existing Security Group) Instance on Public1 and 2
- ELB (application type; select AZ)
- Config Routing (myTarget/healthy threshold:3/add register EC2 instance)

Task #1 :: Demonstration

	AWS Console	Page (demo.html) Access	
EC2		 Instance#1 Public DNS	 Instance#2 Public DNS
ELB		 ELB (application) DNS	

https://ap-northeast-2.console.aws.amazon.com/ec2/v2/home?region=ap-northeast-2#Instances:sort=instan...

	Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status
■	Instance1-po...	i-04cebb265a638901a	t2.micro	ap-northeast-2a	running	2/2 checks ...	None
	Instance2-po...	i-05628a6b1b0674c2f	t2.micro	ap-northeast-2c	running	2/2 checks ...	None

Instance: i-04cebb265a638901a (Instance1-popup) Public DNS: ec2-54-180-9-33.ap-northeast-2.compute.amazonaws.com

Instance ID	i-04cebb265a638901a	Public DNS (IPv4)	ec2-54-180-9-33.ap-northeast-2.compute.amazonaws.com
Instance state	running	IPv4 Public IP	54.180.9.33
Instance type	t2.micro	IPv6 IPs	-
Elastic IPs		Private DNS	ip-10-0-0-31.ap-northeast-2.compute.internal
Availability zone	ap-northeast-2a	Private IPs	10.0.0.31
Security groups	lab-SAASGAPP-1967T0K1JJFP2. view inbound rules. view outbound rules	Secondary private IPs	
Scheduled events	No scheduled events	VPC ID	vpc-03c82cffcef695291 (SAAVPC-popup)
AMI ID	amzn-ami-hvm-2016.09.1.20170119-x86_64-gp2 (ami-dac312b4)	Subnet ID	subnet-069662bc82f740f4d (PublicSubnetA-popup)
Platform	-	Network interfaces	eth0
IAM role	-	Source/dest. check	True
Key pair name	-	T2/T3 Unlimited	Disabled
Owner	174186896117	EBS-optimized	False
Launch time	July 2, 2019 at 7:30:52 AM UTC+9 (less than one hour)	Root device type	ebs

https://ap-northeast-2.console.aws.amazon.com/ec2/v2/home?region=ap-northeast-2#LoadBalancers:sort=l...

Load balancer: lab-SAAELB-O51JHGT5PDVO

Basic Configuration

Name	lab-SAAELB-O51JHGT5PDVO
ARN	arn:aws:elasticloadbalancing:ap-northeast-2:174186896117:loadbalancer/app/lab-SAAELB-O51JHGT5PDVO/0402323f9a638884
DNS name	lab-SAAELB-O51JHGT5PDVO-929068282.ap-northeast-2.elb.amazonaws.com (A Record)
State	active
Type	application
Scheme	internet-facing
IP address type	ipv4
VPC	vpc-03c82cffcef695291
Availability Zones	subnet-069662bc82f740f4d - ap-northeast-2a IPv4 address: Assigned by AWS; subnet-0f146bd59787c6fad - ap-northeast-2c IPv4 address: Assigned by AWS
Hosted zone	ZWKZPGTI48KDX
Creation time	July 2, 2019 at 7:30:54 AM UTC+9

Security

Security groups: sg-0a0e4939c55369a7a, lab-SAASGELB-1X3R956RSW700

Task#2

Task #2 :: Requirements, 1/2

HA/Reliable/Robust

- A Highly available architecture that resists to the failure of single component
- Disaster Recovery should be considered in case of multiple components failure
- A self-healing infrastructure that recovers from failed service instances
- Cost-effectiveness should also be considered across all components of the architecture

High Performance

- Configure their database and data access layer for high performance and throughput
- Browser very low latency even though a large portion of their user base will be from far away
- Effective distribution of load regardless whether it’s http/1.1 or http/2.0 request

Task #2 :: Requirements, 2/2

Security

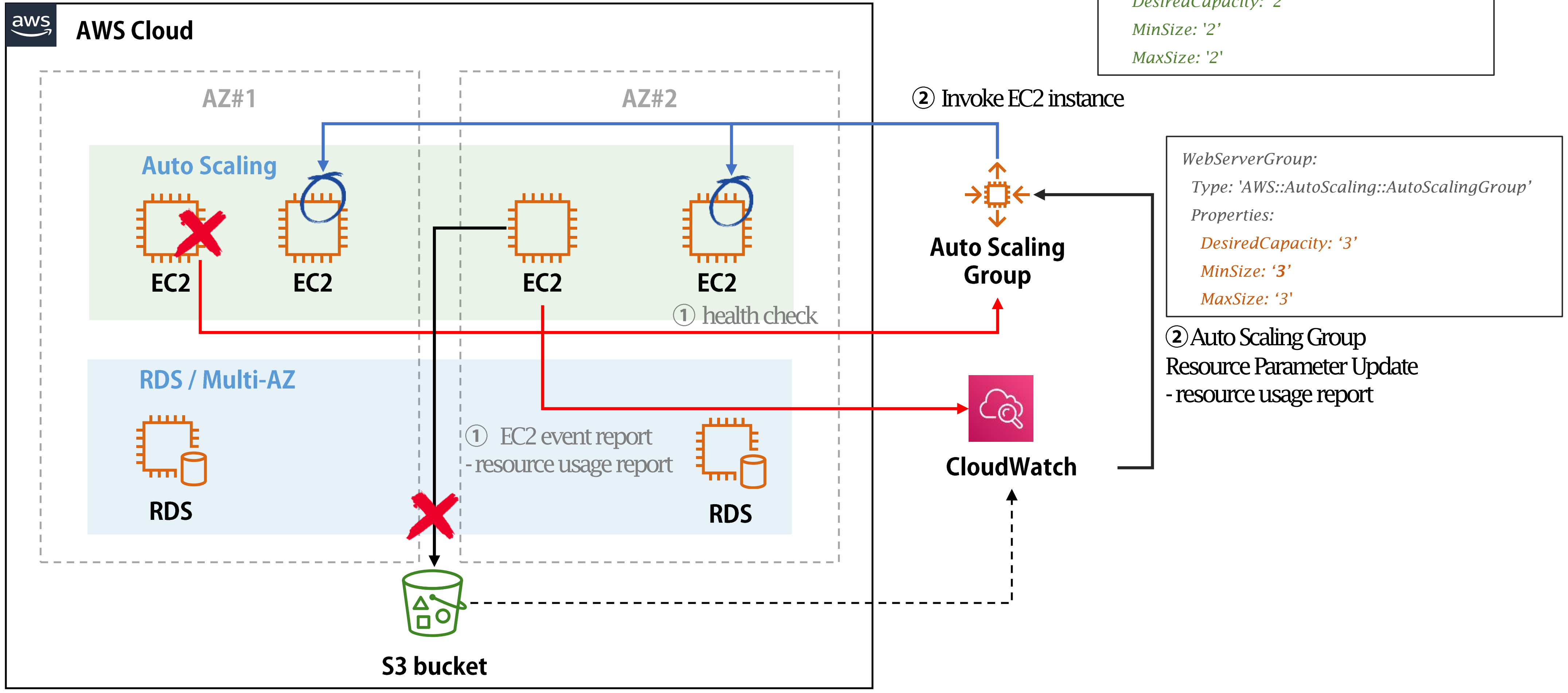
- Security of data at rest and in transit
- Securing access to the environment as the delivery team expands

DevOps, Operation/Maintenance

- An archival strategy for inactive objects greater than 6 months
- Ability to easily manage and replicate multiple environments based on their blueprint architecture.
- Application lifecycle management should be considered as a DevOps strategy
- Cost-effectiveness should also be considered across all components of the architecture
- Access logs generated need to be collected and aggregated for visualization
- Scaling to meet the demand, but with uncertainty around when and how much this demand ..

Task #2 :: HA/Reliable/Robust

- Design redundant architecture (n+0)
- Using dynamic autoscaling architecture w/ event audit

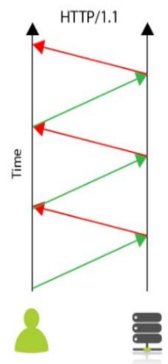
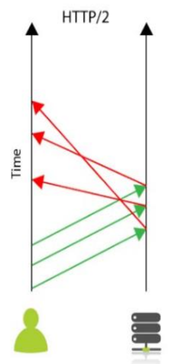


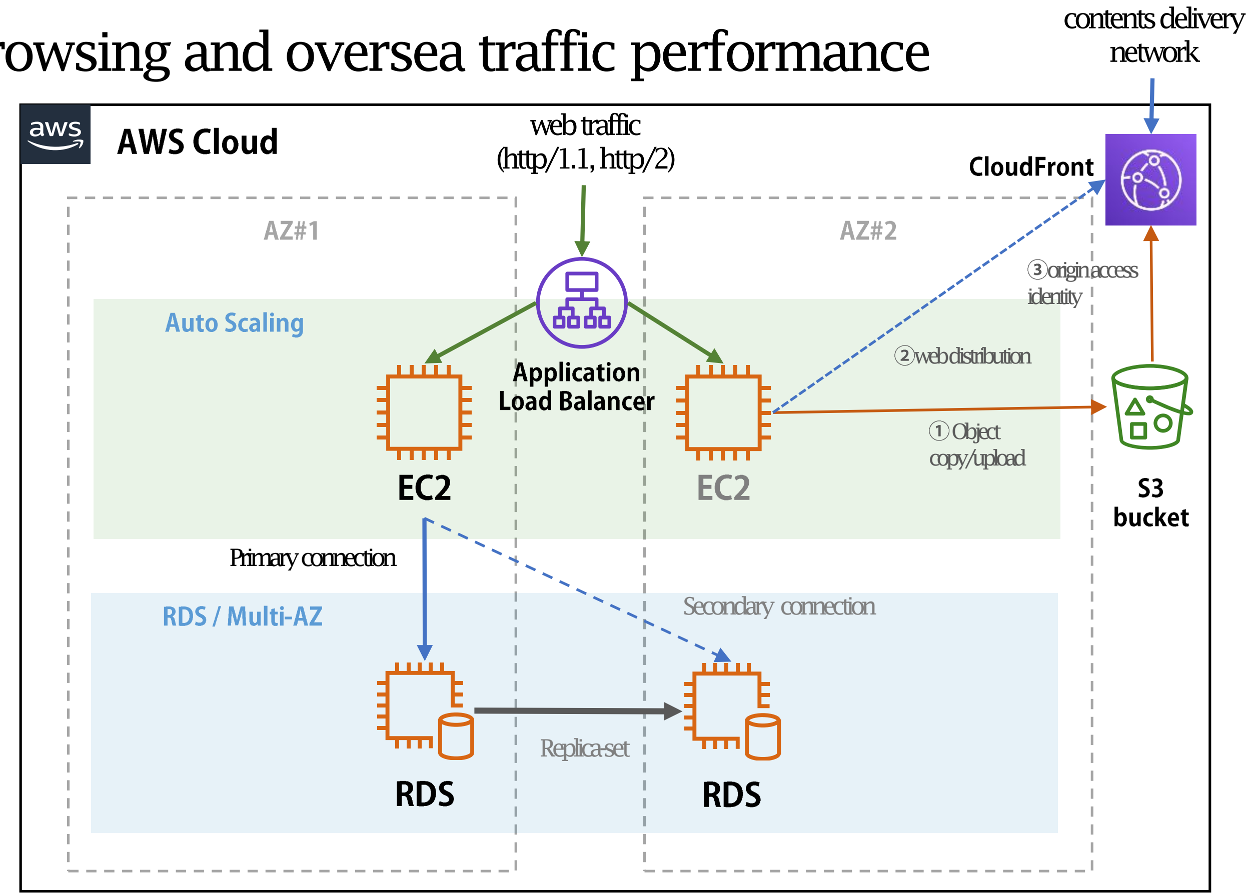
```
WebServerGroup:
  Type: 'AWS::AutoScaling::AutoScalingGroup'
  Properties:
    DesiredCapacity: '2'
    MinSize: '2'
    MaxSize: '2'
```

```
WebServerGroup:
  Type: 'AWS::AutoScaling::AutoScalingGroup'
  Properties:
    DesiredCapacity: '3'
    MinSize: '3'
    MaxSize: '3'
```

Task #2 :: High Performance

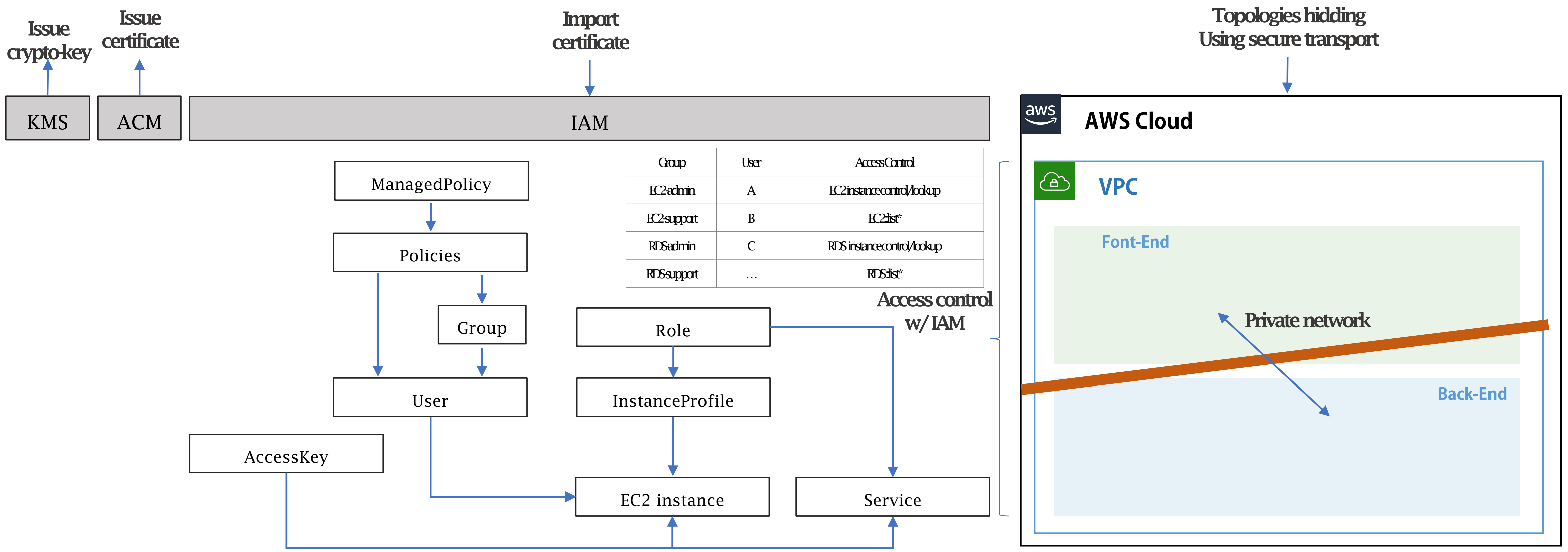
- Using ELB(application) → Effective distribution of L7 load (lookup application payload feature)
- RDS/Multi-AZ support scale-out DB instance for performance (internal sharding and replica)
- CloudFront makes a advanced low latency browsing and oversea traffic performance

Versionc	HTTP/1	HTTP/1.1	HTTP/2	HTTP/3
Transport	TCP	TCP	TCP	UDP(?)
Persistent connection	x	o	o	-
Overhead - HOL blocking - RTT increment - Low latency	o	o	X Image sprite, domain sharding, css/js compress, Multiplexed Streams	
Procedure			 images: https://bit.ly/2XJbxJY	
ELB type		Network/L4	Application/L7	UDP support



Task #2 :: Security

- Separate all of network-entity
- Build a own VPC, isolate between front and back-end nodes, using secure transport (SSL/HTTPS)
- Enforce a user access and authorization w/ hierarchy of control thuru role/group/policy
- Securing access to the environment as the delivery team expands, IAM



Group	User	AccessControl
EC2-admin	A	EC2 instance control/lookup
EC2-support	B	EC2:list*
RDS-admin	C	RDS instance control/lookup
RDS-support	…	RDS:list*

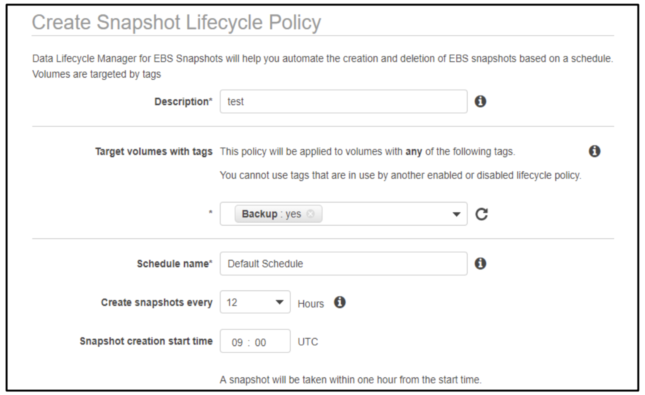
Task #2 :: DevOps, Operation/Maintenance

- Managing a Lifecycle policy for an S3 bucket (archiving for inactive objects > 6 months)
- and EBS backup image-snapshot, application beanstalk.

Archiving <6 months S3 Object to GLACIER Storage class Examples

```
<LifecycleConfiguration>
<Rule>
   <ID>Transition and Expiration Rule</ID>
   <Filter> <Prefix>tax/</Prefix> </Filter>
   <Status>Enabled</Status>
   <Transition>
      <Days>180</Days>
      <StorageClass>GLACIER</StorageClass>
   </Transition>
   <Expiration> <Days>365*5</Days> </Expiration>
</Rule>
</LifecycleConfiguration>
```

Create EBS snapshot lifecycle policy



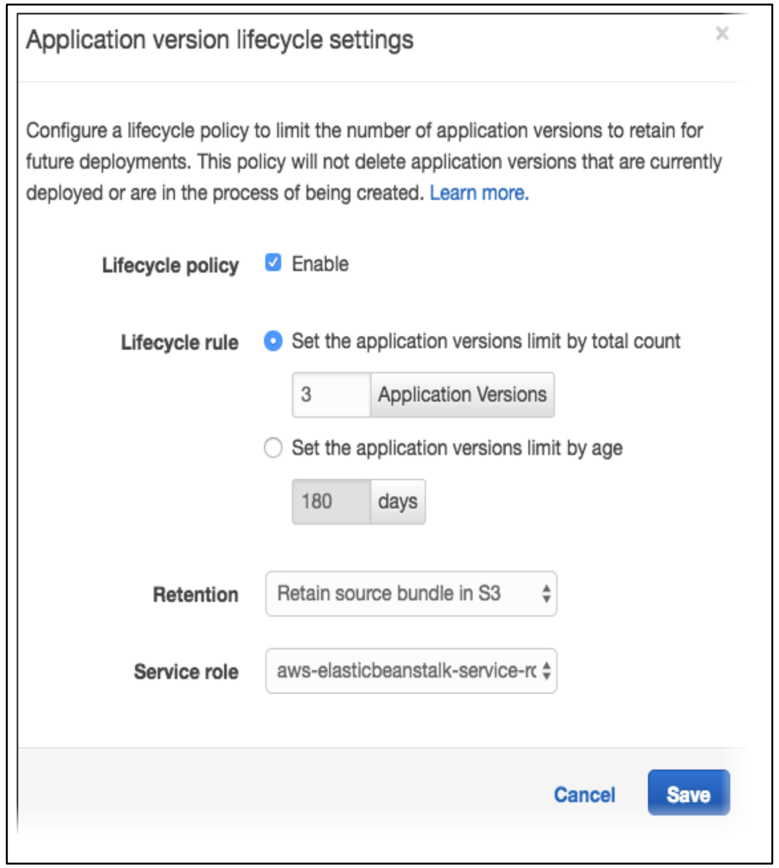
Create Snapshot Lifecycle Policy

Data Lifecycle Manager for EBS Snapshots will help you automate the creation and deletion of EBS snapshots based on a schedule. Volumes are targeted by tags

Description*	test
Target volumes with tags	This policy will be applied to volumes with **any** of the following tags. You cannot use tags that are in use by another enabled or disabled lifecycle policy.
*	**Backup** : yes
Schedule name*	Default Schedule
Create snapshots every	12 Hours
Snapshot creation start time	09 : 00 UTC

A snapshot will be taken within one hour from the start time.

Beanstalk Application, lifecycle setting



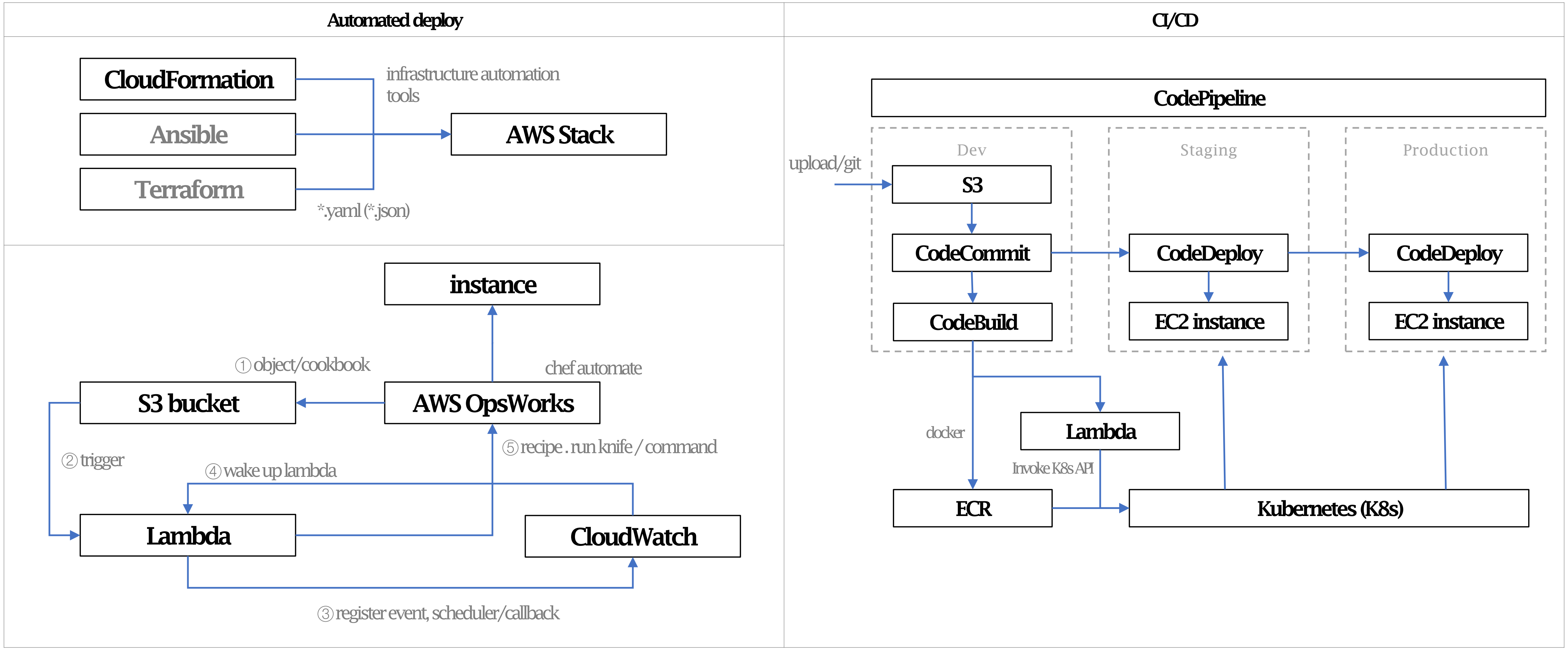
Application version lifecycle settings

Configure a lifecycle policy to limit the number of application versions to retain for future deployments. This policy will not delete application versions that are currently deployed or are in the process of being created. Learn more.

Lifecycle policy	☑ Enable
Lifecycle rule	● Set the application versions limit by total count
	3 Application Versions
	○ Set the application versions limit by age
	180 days
Retention	Retain source bundle in S3
Service role	aws-elasticbeanstalk-service-rc

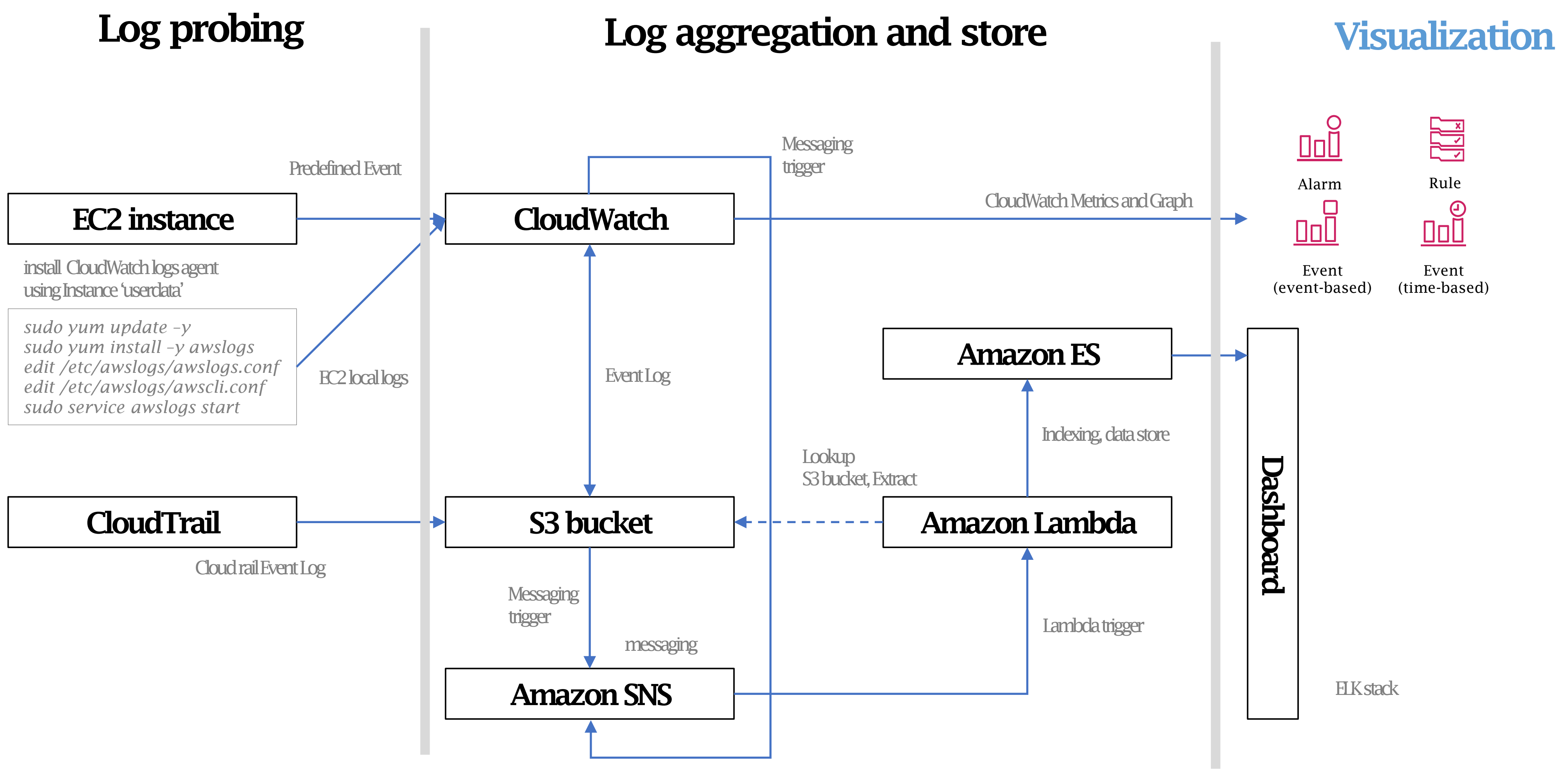
Task #2 :: DevOps, Operation/Maintenance

- Using diverse deploy method exists.



Task #2 :: DevOps, Operation/Maintenance

- Access logs generated need to be collected and aggregated for visualization



```
sudo yum update -y
sudo yum install -y awslogs
edit /etc/awslogs/awslogs.conf
edit /etc/awslogs/awscli.conf
sudo service awslogs start
```

Conclusion

Conclusion :: At a glance, our customer

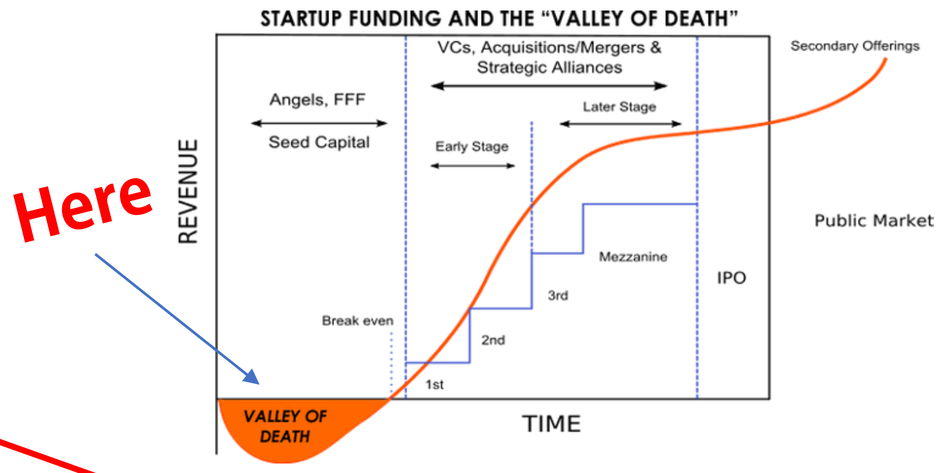
I was a freelance cloud architect, so I met and talked my customer for finding their real needs and problems.

→Customer does not have much money, but want to build and operation on cloud

Customer persona-model

- Start-up company (founded in less than 1 years)
- Proof of concept (PoC) stage, but want a deploy on AWS cloud
- Full-time employee: 4~5
- Stock holder's equity: ~ $ 250,000 (But, most money is used as employee salary)

< $ 1,500 / monthly



What they really want?

- Cost-effective cloud model, monthly subscribe model.
- Initially, they want a more cheaper architecture model, but they will be aimed a scalability and global region
- They want to know exactly how much money they have to spend. on their business plan.

Conclusion :: guessing, our customer service

My customer prepare the mobile media service that needs a API servers and databases on cloud. In order to show the service to VC / investor for seed money, at least 10,000 users and using 5 time in daily per should be secured.

→ So, I have a simulation traffic and capacity for user service. Then, induced below results.

. Service Capa. Simulation : Instance Selection Guiding: Instance-type: Large type / 340GiB EBS

[1] User	10,000	
[2] Service Attempt (per user)	0.2 /hour	
[3] Mean hold time	200 sec	
[4] Concurrent Service	110/sec	[1]*[2]*[3]/60/60
[5] Transaction	670	[4]*6
[6] tpmC	52,000	[5]*60*130%
[7] Iops	2,680	[4]*4
[8] Storage Volume	340 GiB	EBS / Iops ratio

Instance	#vCPU	tpmC(e)
Small	1	42,253
Large	2	82,506
Xlarge	4	165,012
2xlarge	8	330,024
4xlarge	16	660,048

Reference HW: large/2.4GHz intel/xeon® E5-2676 v3(Haswell)

Conclusion :: Proposal architecture

My customer needs a cost-effective cloud architecture (endure 10,000 user service and less money to pay monthly)

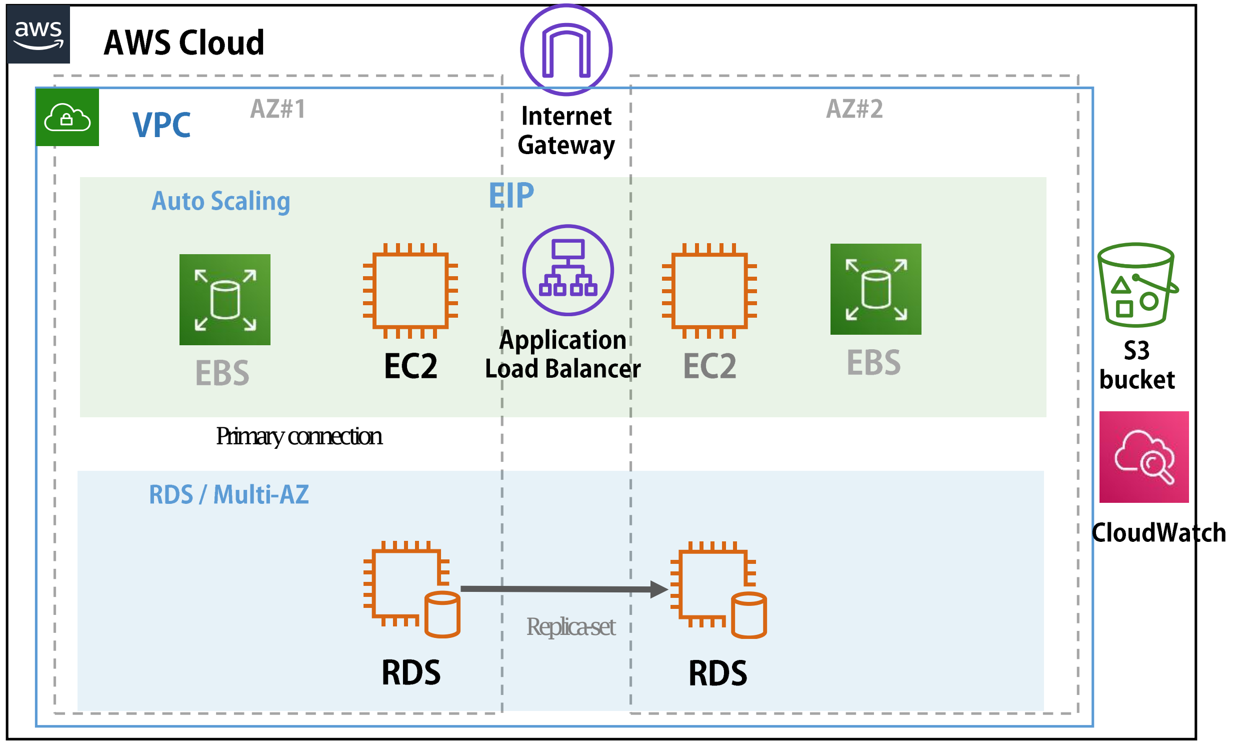
→ So, I propose minimal-size cloud architecture. but, it was solving the reliability, security, cost, operation and performance needs.

Seoul Region (ap-northeast-2)

model	Unit	Cost	Description
EC2 (t3.xlarge)	2	$ 609	
EBS (340GB)	2		
EIP	1		
S3 (1000GB)	1		
ELB/Application	1		
RDS /MySQL/Multi AZ (db.t2.large)	2		
CloudWatch (5 x 5, 1min)	1		
IAM			
Total		$ 1,383	< $1,500

Cost-effective AWS cloud architecture

- $1,383 < Expected budget = $ 1,500



Conclusion :: Explain the propose architecture

I derived a some keyword for facing user problem and needs (improve the reliability, security, cost, operation and performance before the project goes into production) previous chapter.

- Architecture (Redundancy, Distributed model) for HA
- Load balancing for application level
- Consider scale-up and scale-out
- Network virtualization, isolation, crypto for data and transit security
- IAM hierarchy scheme apply for diverse delivery team
- Easy deploy, operation and support visualization

Conclusion :: Scaling meet the customer demands

Considerations	Service	Using/Scale-out Checkpoint	Add on service (for large-scale and global biz)
HA architecture (for zero downtime)	Dual EC2 (+EBS) w/ auto-scaling S3 RDS	When increasing user and traffic (2,600 tpmC /+EC2)	Consider DB Clustering
Resist to regional failure	Using different regions (VPC and RDS/Multi-AZ)		
Support security	Using HTTPS (with certification) Network virtualization (VPC) and topology hidding with public and private subnet segmentation. Access-control with IAM (prepare policy and group, then allocate group to user/delivery team)	Each service policy and permission divided 3 grade with admin / operation (start/stop)/ support (view)	
Performance	Using Application LoadBalancer (some latency exist) Apply service grouping (like RDS/multi-AZ, autoscale)		CloudFront
Operation / Visualization	using AWS admin console and CloudWatch dashboard (monitoring component event – predefined and save to S3 bucket) Using CloudWatch agent logs on EC2 (pre-install)		Lambda SNS ES

Conclusion :: Scaling meet the customer demands

Considerations	Service	Using/Scale-out Checkpoint	Add on service (for large-scale and global biz)
instance/object/data life cycle management	Using AMI and snapshot service for EC2 S3 bucket life-cycle mgmt.' EBS backup schedule	Daily backup Expired S3 bucket objects will be archiving and removed	
DR (backup and restore)	Restore EC2 w/ AMI or backup snapshot Restore backup EBS		
Blueprint deploy	Using CloudFormation(+ansible) for infrastructure		OpsWorks Beanstalk
CI/CD	Manually deploy (using git tool)		CodePipeline (CodeCommit, CodeDeploy) ECR K8s