

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

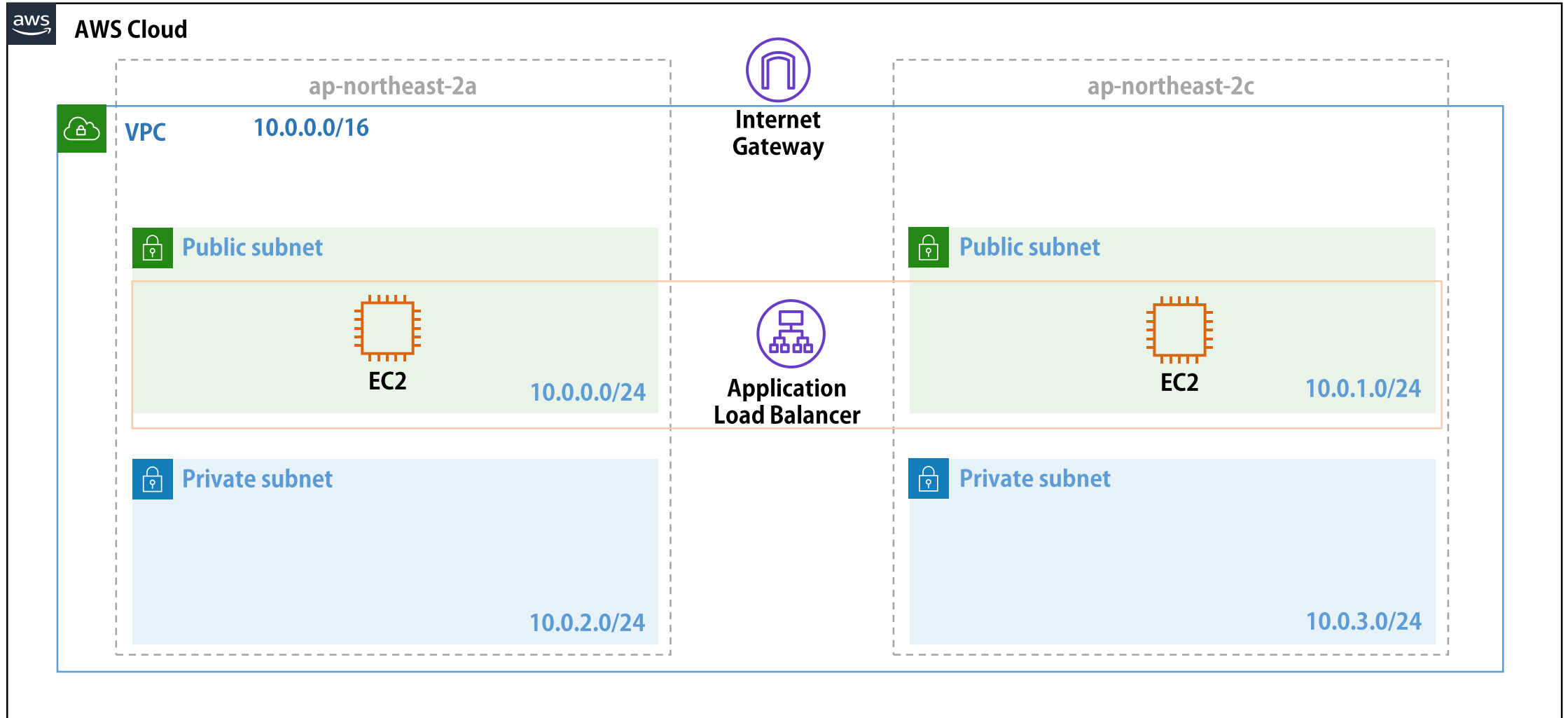
- Trouble shoot the issue by modifying the misconfiguration generated by given CloudFormation template and load demo.html page thru ELB hostname

Task #2

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

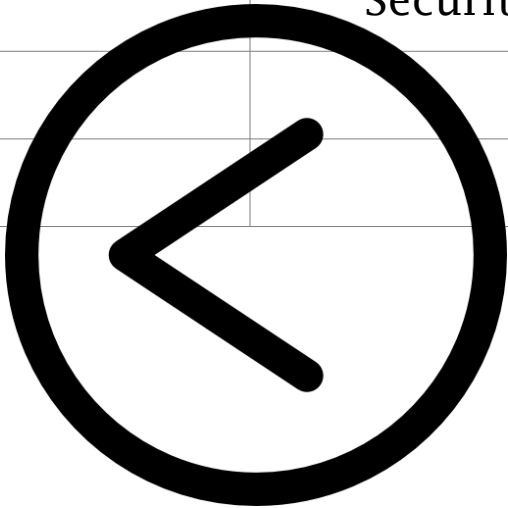
Task#1

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)

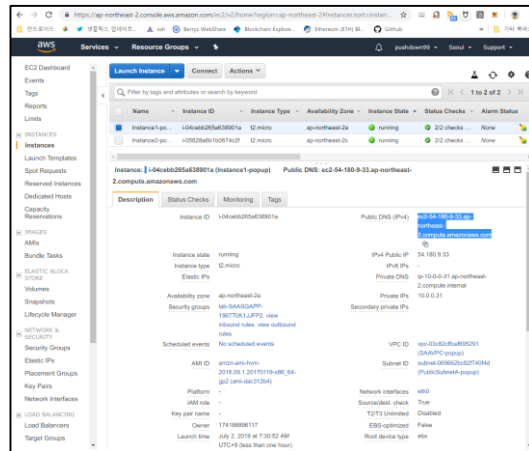
Launch a EC2 and ELB/ALB

- 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

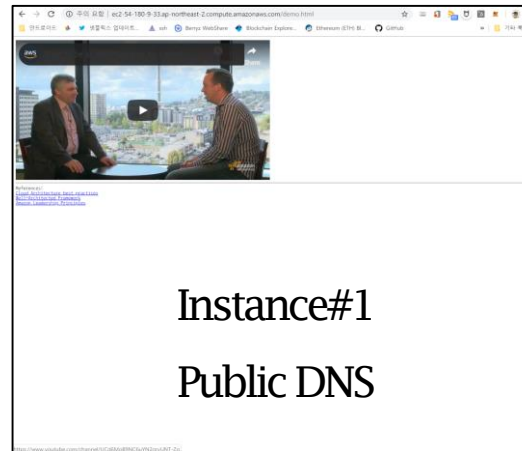
AWS Console

EC2

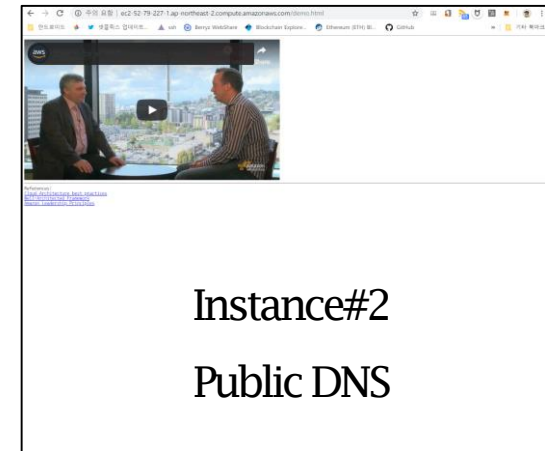


Page (demo.html) Access

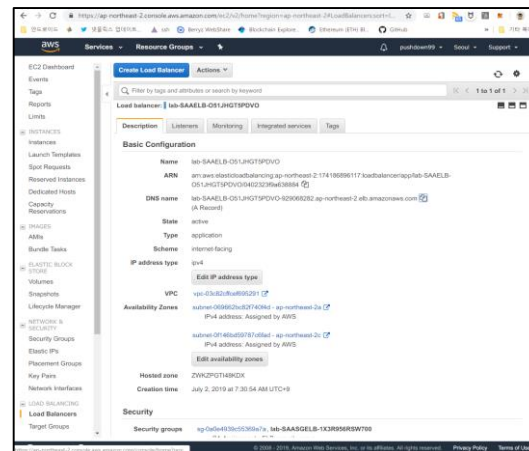
Instance#1
Public DNS



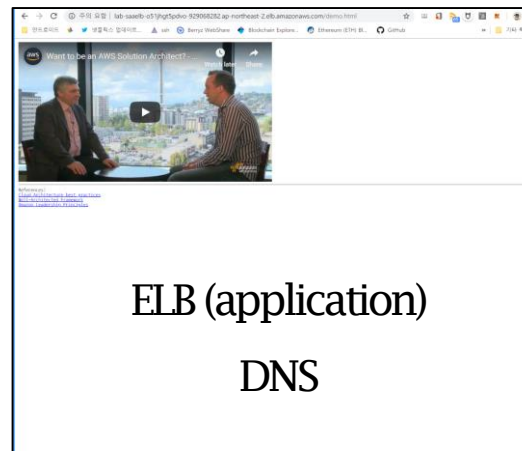
Instance#2
Public DNS



ELB



ELB (application)
DNS



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

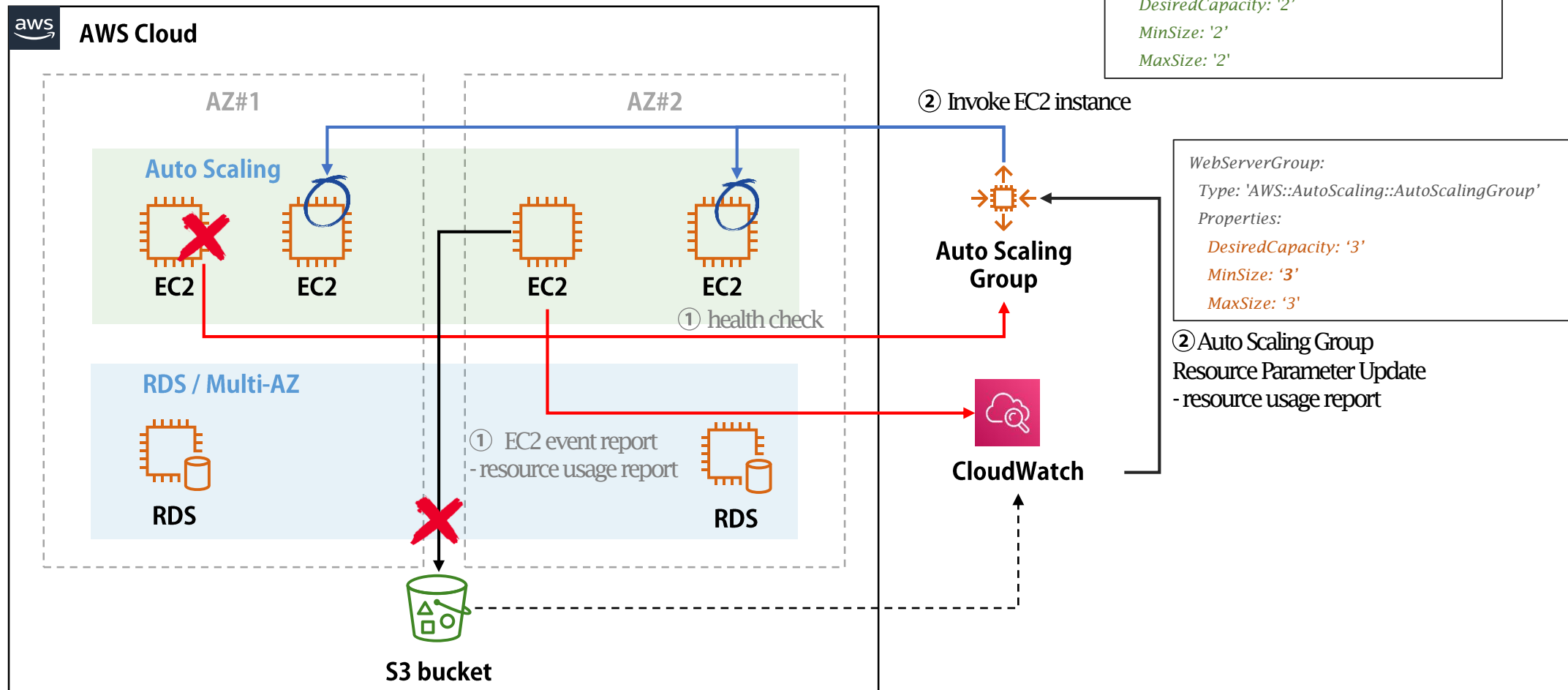
- Security of data at rest and in transit
- Securing access to the environment as the delivery team expands

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 ..

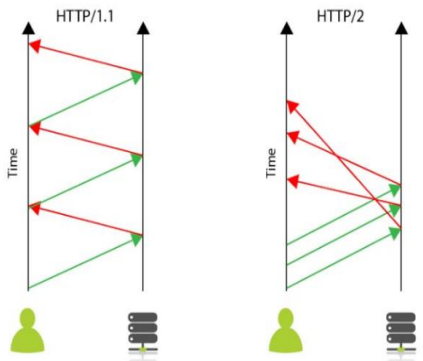
Task #2 :: HA/Reliable/Robust

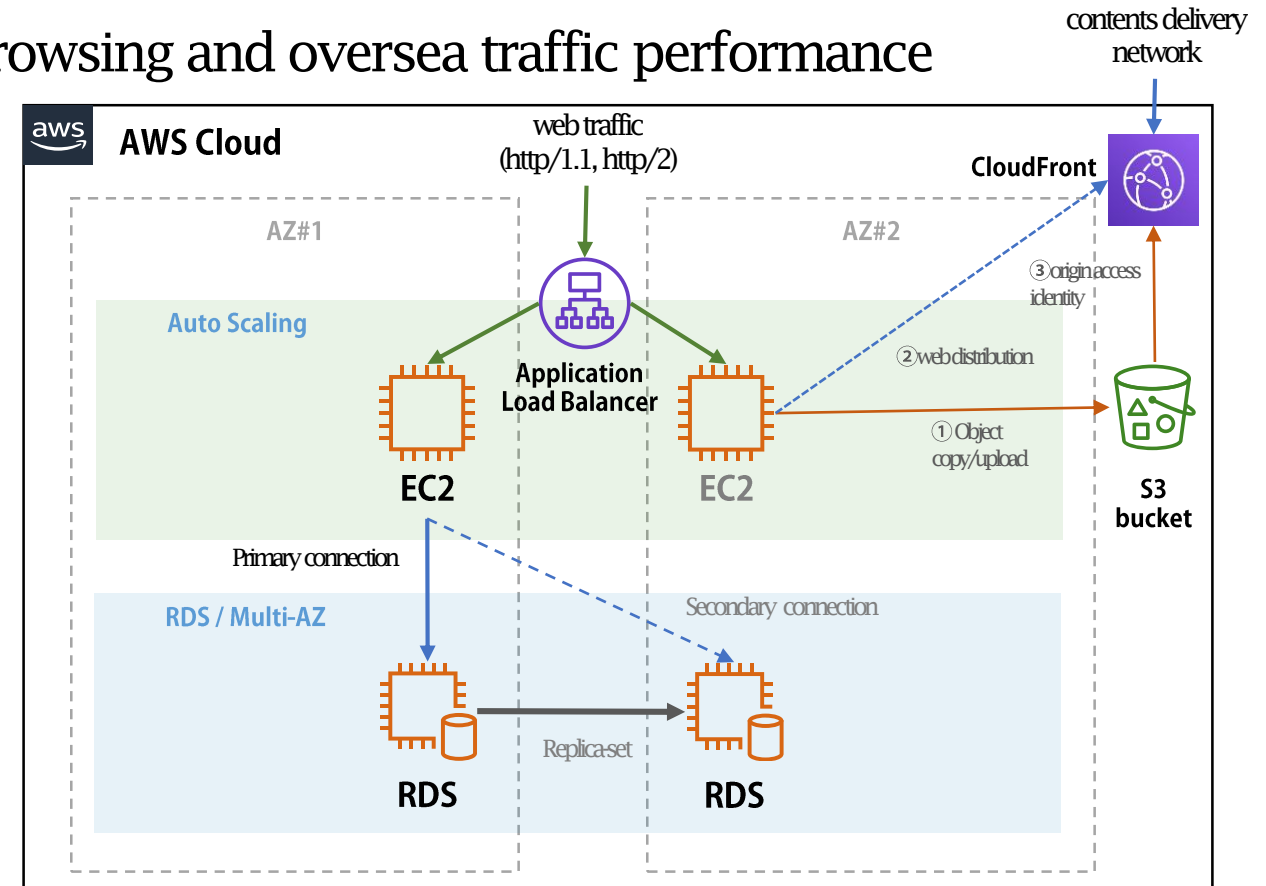
- Design redundant architecture (n+0)
- Using dynamic autoscaling architecture w/ event audit



Task #2 :: High Performance

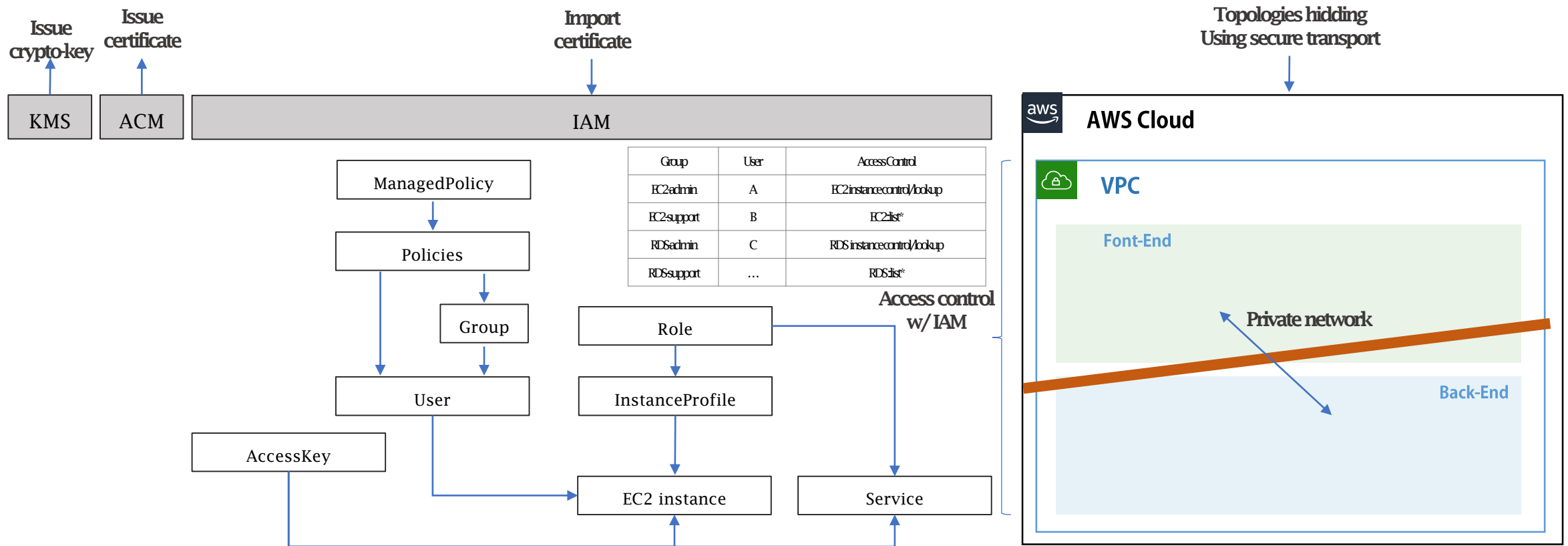
- Using ELB(application) → Effective distribution of L7 load (lookup application payload feature)
- RDS/Multi-AZ support scale-out DB instance for performance (internal sharding and replica)
- CloudFront makes a advanced low latency browsing and oversea traffic performance

Version	HTTP/1	HTTP/1.1	HTTP/2	HTTP/3
Transport	TCP	TCP	TCP	UDP(?)
Persistent connection	x	o	o	-
Overhead <ul style="list-style-type: none"> - HOL blocking - RTT increment - Low latency 	o	o	X Image sprite, domain sharding, css/js compress, Multiplexed Streams	
Procedure		 <p>images: https://bit.ly/2XJbxjY</p>		
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 thru role/group/policy
- Securing access to the environment as the delivery team expands



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* ⓘ

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.

* ⓘ

Schedule name* ⓘ

Create snapshots every Hours ⓘ

Snapshot creation start time 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

Application Versions

☐ Set the application versions limit by age

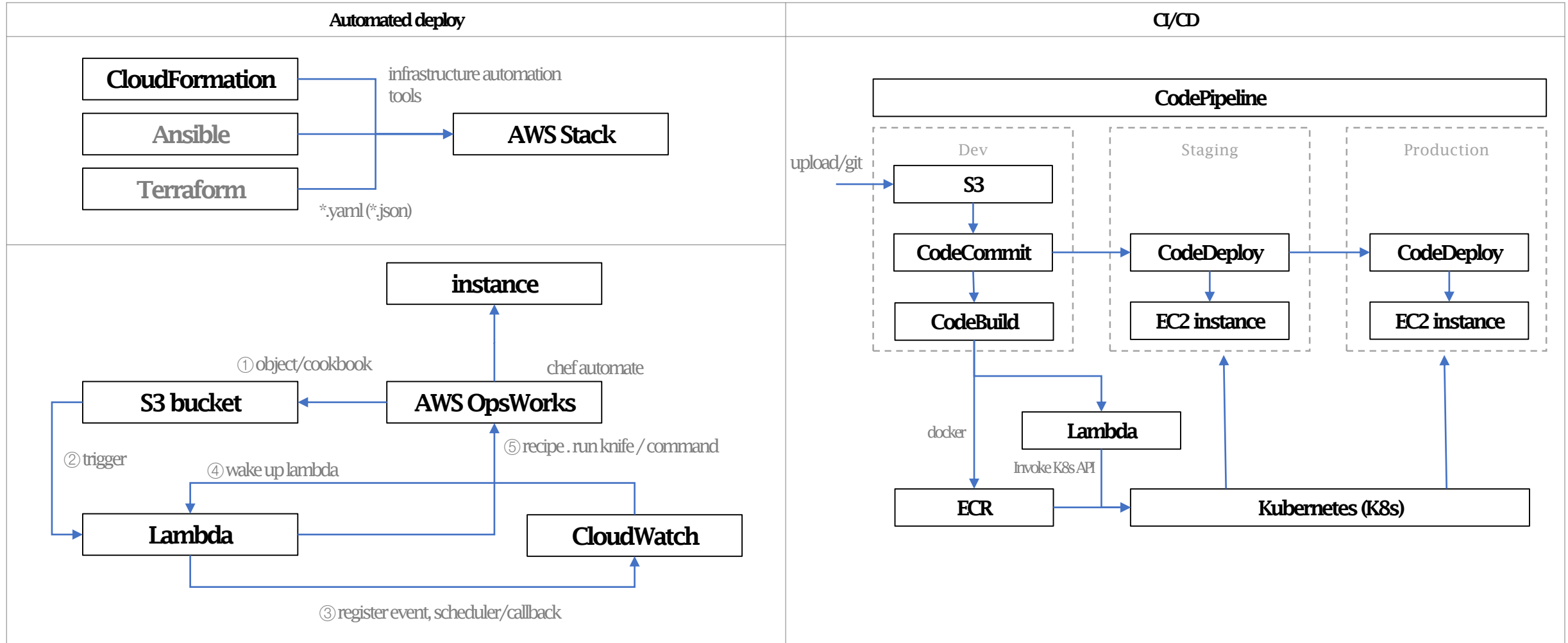
days

Retention

Service role

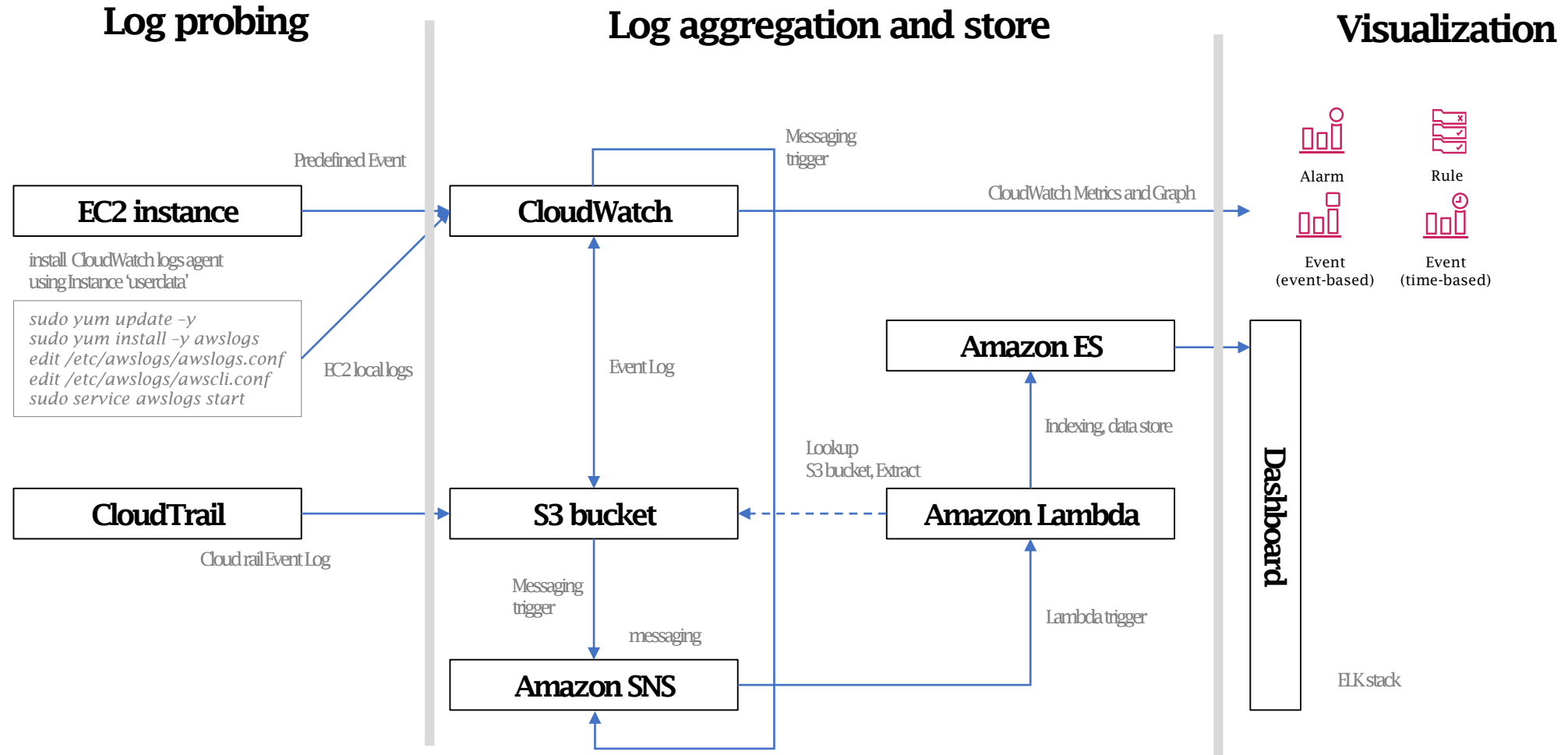
Task #2 :: DevOps, Operation/Maintenance

- Using diverse deploy method exists.



Task #2 :: DevOps, Operation/Maintenance

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

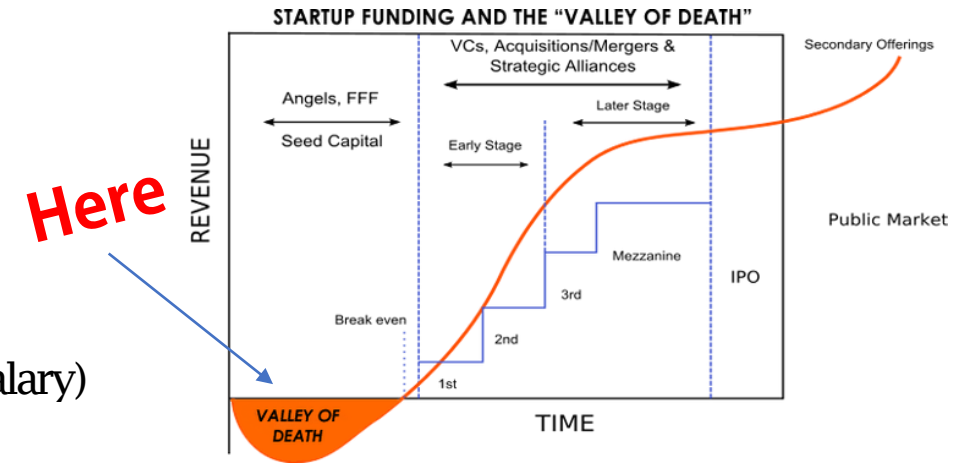


Conclusion

Conclusion :: At a glance, our customer

Persona

- 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)



What they really want?

< \$ 1,500 / monthly

- **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

Ex) Service Capa. Simulation

[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)

Instance Selection Guide.

- Instance-type: Large type / 340GiB EBS

Conclusion :: Proposal

My Proposal,

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		
Total		\$ 1,383	< \$ 1,500

Cost-effective AWS cloud architecture

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

