
AWS
re:Invent

CON215

Samsung Smart TV: Globally scaling with AWS Fargate Spot

June Park
Solutions Architect
AWS

Junhyeok Song
Staff Engineer
Samsung Electronics
Visual Display Division

Kyungsik Myung
Engineer
Samsung Electronics
Visual Display Division

AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Agenda

- Samsung SmartTV service
- Migration journey to AWS
- AWS Fargate / Fargate Spot
- AWS Cloud Development Kit (AWS CDK)
- Cost optimization
- Summary



© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Samsung Smart TV

Age of experience

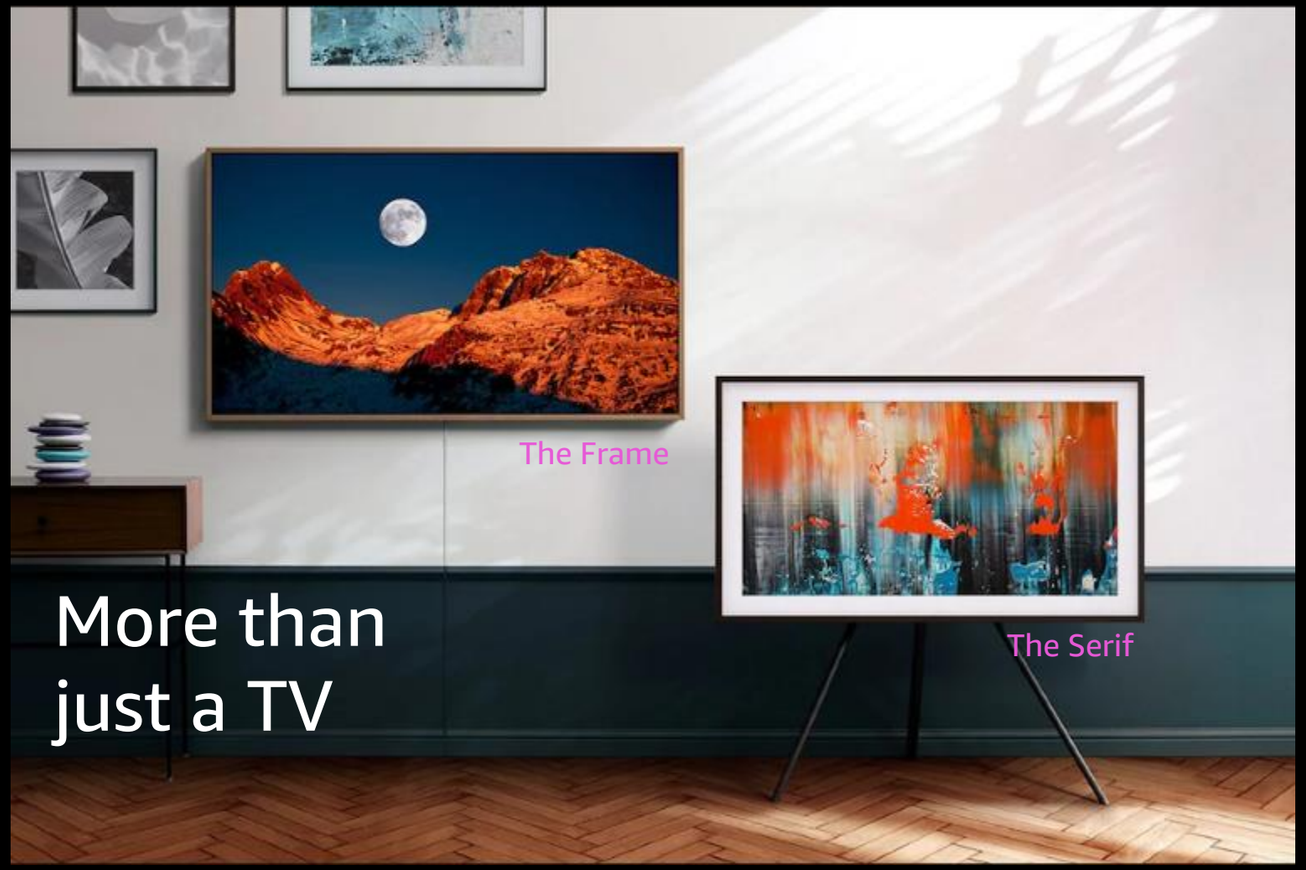


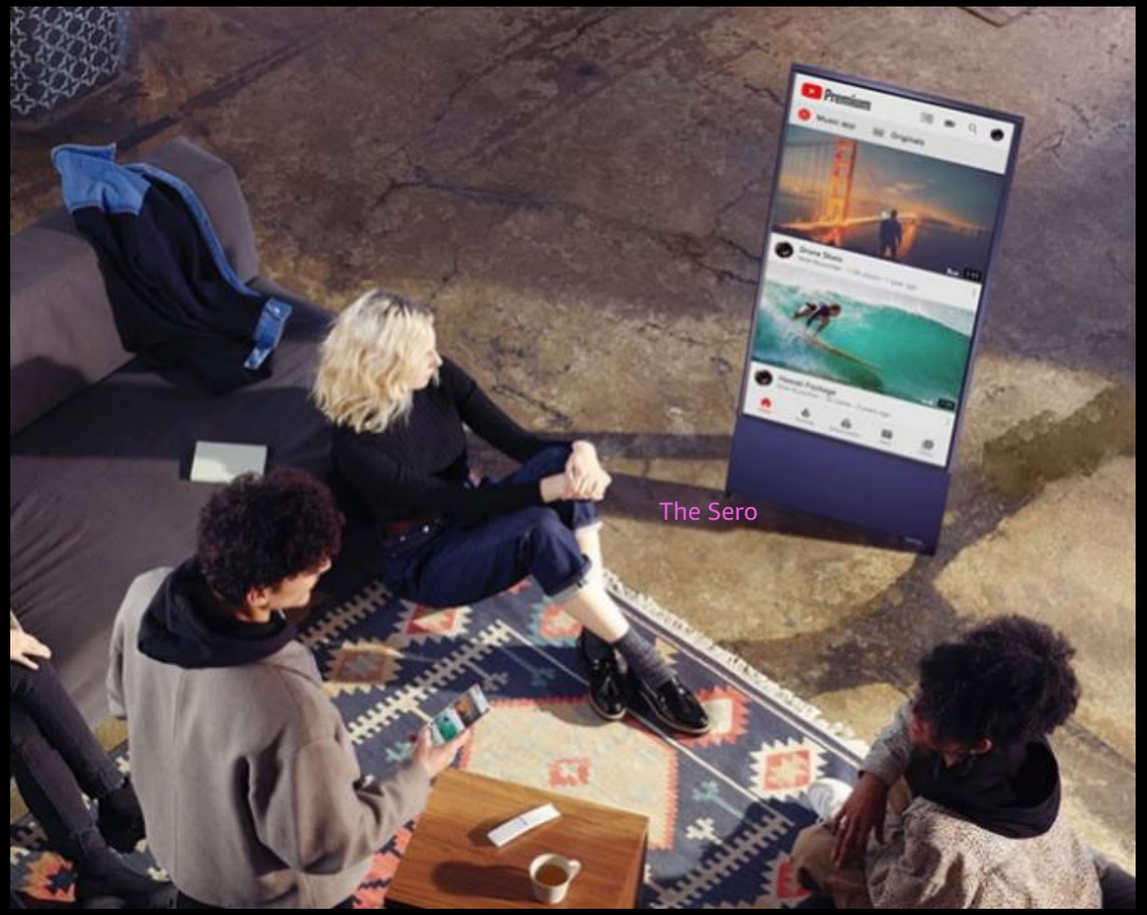
AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Lifestyle TV

MORE THAN JUST A TV





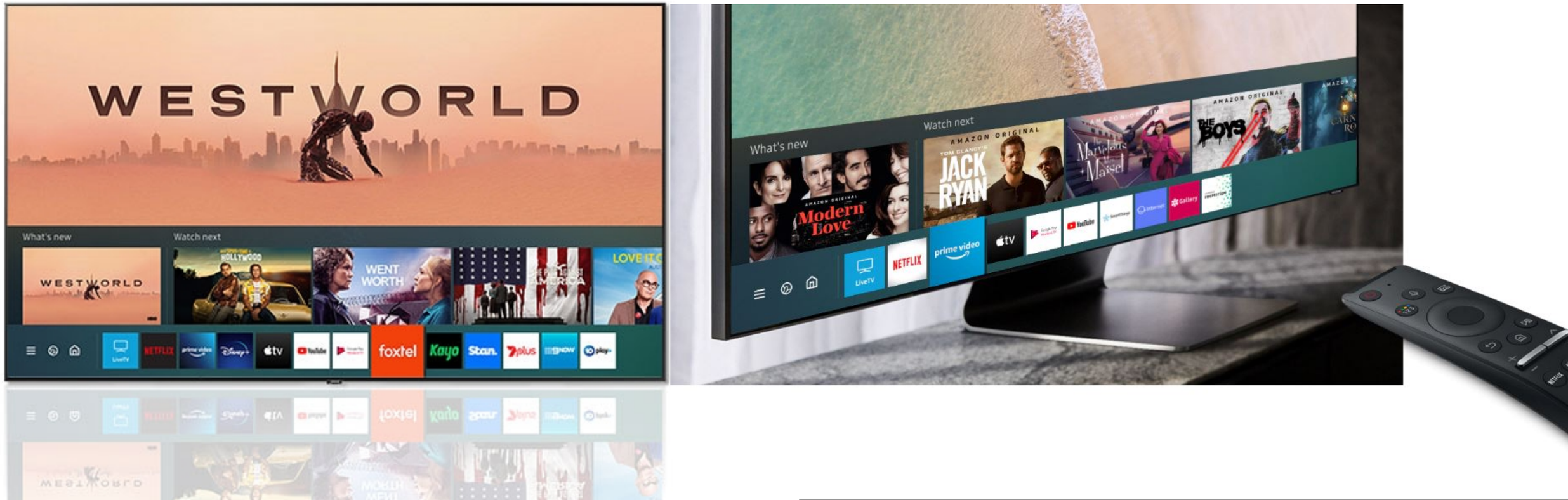
Change the way you see your TV. Samsung’s 2020 Lifestyle Range TVs are so much more than screens, they're stylish pieces of design that’ll elevate your space.

Frame, Serif, Sero, and Terrace

Smart TV Powered by TIZEN

Samsung has maintained its no. 1 position in the global TV market for 14 consecutive years

140 million TIZEN TVs are connected to the internet from 197 countries



Samsung services

TV PLUS





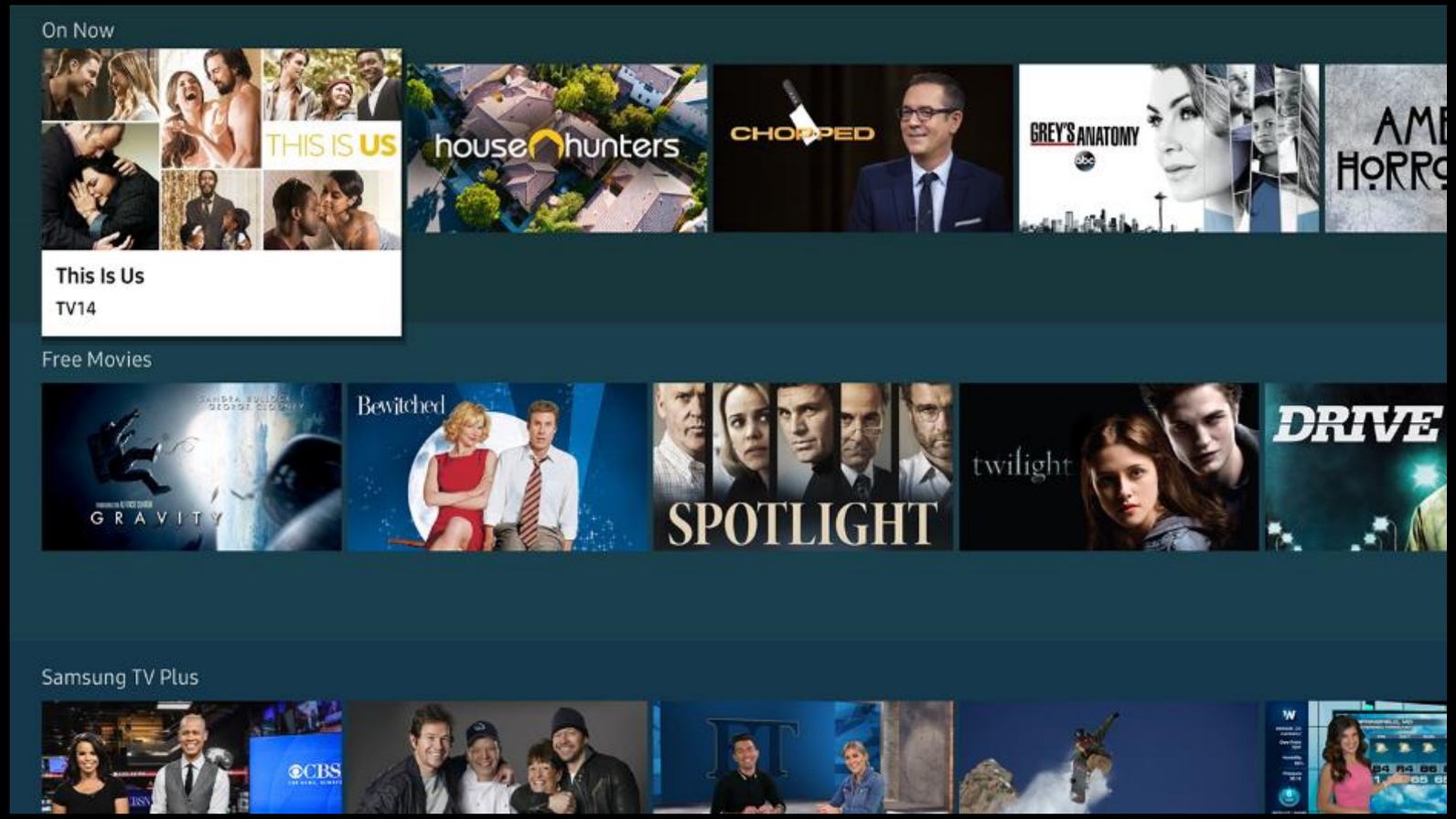
Free video streaming service experienced by channel
Enjoy great content of the most popular genres

Samsung services

UNIVERSAL GUIDE



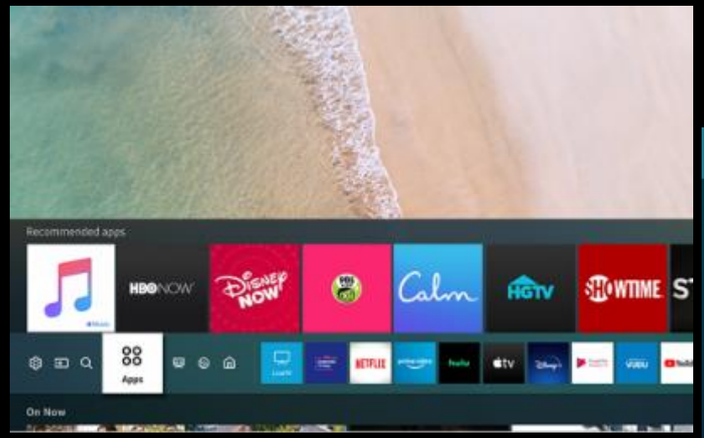
People spent 50 minutes per day to search for content



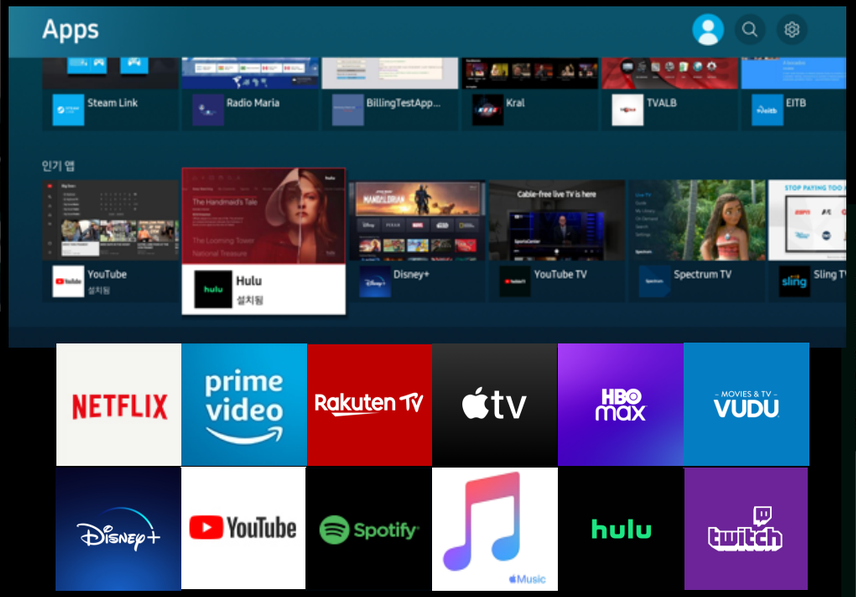
Content discovery service that matches user preference

APPs

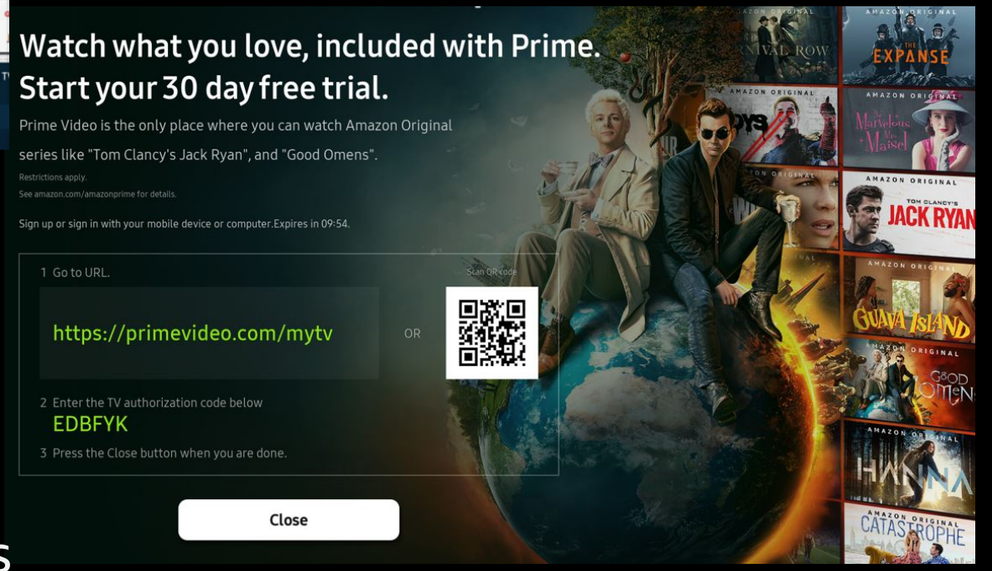
PLATFORM SERVER FOR APP CONTENT DISTRIBUTION



SmartHub Home



TV App Store



OOBE Promotion

Apps service has launched in 2010 serving trillions of requests per month from all over the world

OOBE: Out of Box experience

Globally scaling with AWS Fargate Spot



AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Background

WHERE WE WERE

- Monolithic application for 2010–2017 Smart TV products
- Single massive database in on-prem IDC
- Database scaling issues and cost overheads
- Direct lift and shift is not possible (* Failed in 2013)
- Sudden traffic spike during global app launch

Background

WHAT WE HAVE LEARNED FROM FAILURES OF TWO MIGRATION ATTEMPTS

We realized that migration is not easy for large massive scale services

- Requires architectural change
- Need to carefully plan for migration in multiple phases due to its size and reduce service downtime

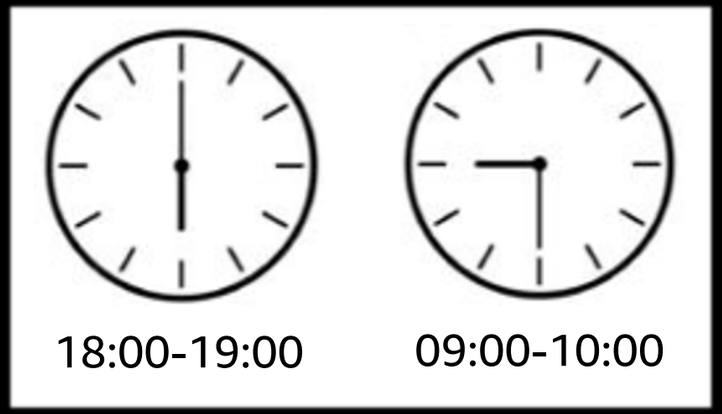
Amazon Elastic Compute Cloud (Amazon EC2) and scaling performance was not able to fully support our system

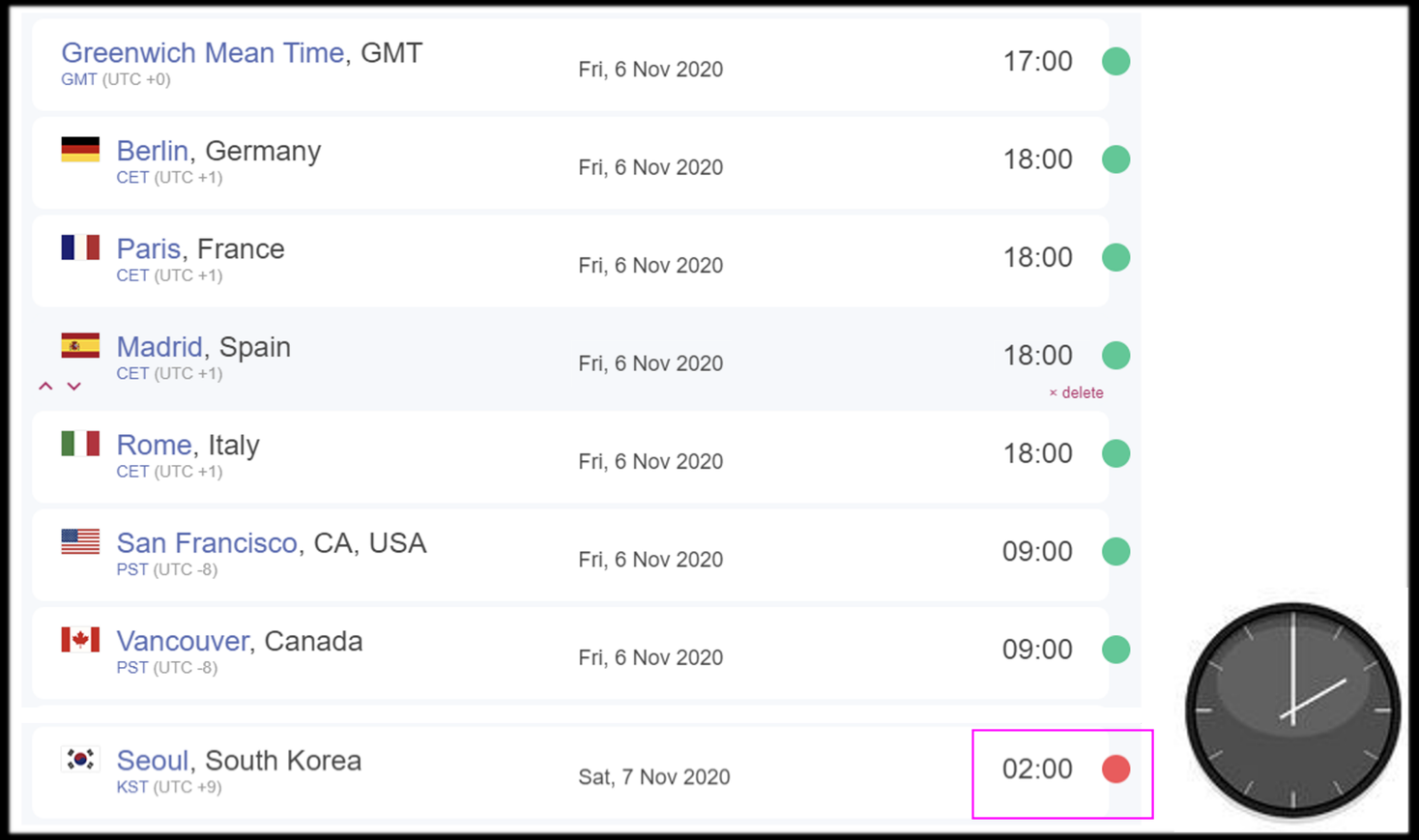
- Scale of API transactions was too vast for any combination of Amazon EC2 Auto Scaling to support
- Database transactions were too massive for Amazon Relational Database Service

Background

TIMEZONE

- When do you watch your TV?





Peak time for TV viewing from other parts of world is equivalent to 2 AM KST and developers have to handle alarms at 2 AM if there is any issue

Background

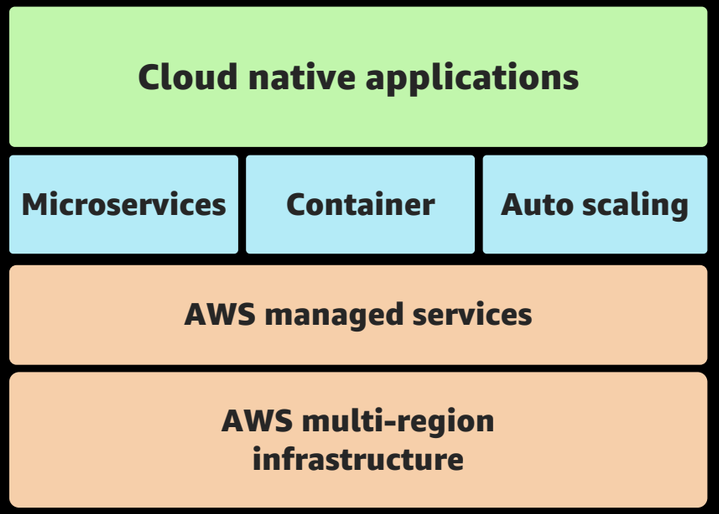
TIME ZONE

- Business continuity is critical
- BUT, so is developer’s life {work and life needs to be balanced!}
- No one likes to receive an alarm and phone call at 2 AM due to a system failure



Approaches

GOAL



- Cloud native and microservices on managed service
- Goodbye to the IDC

Goal – focus on the business logic, not infrastructure and monitoring

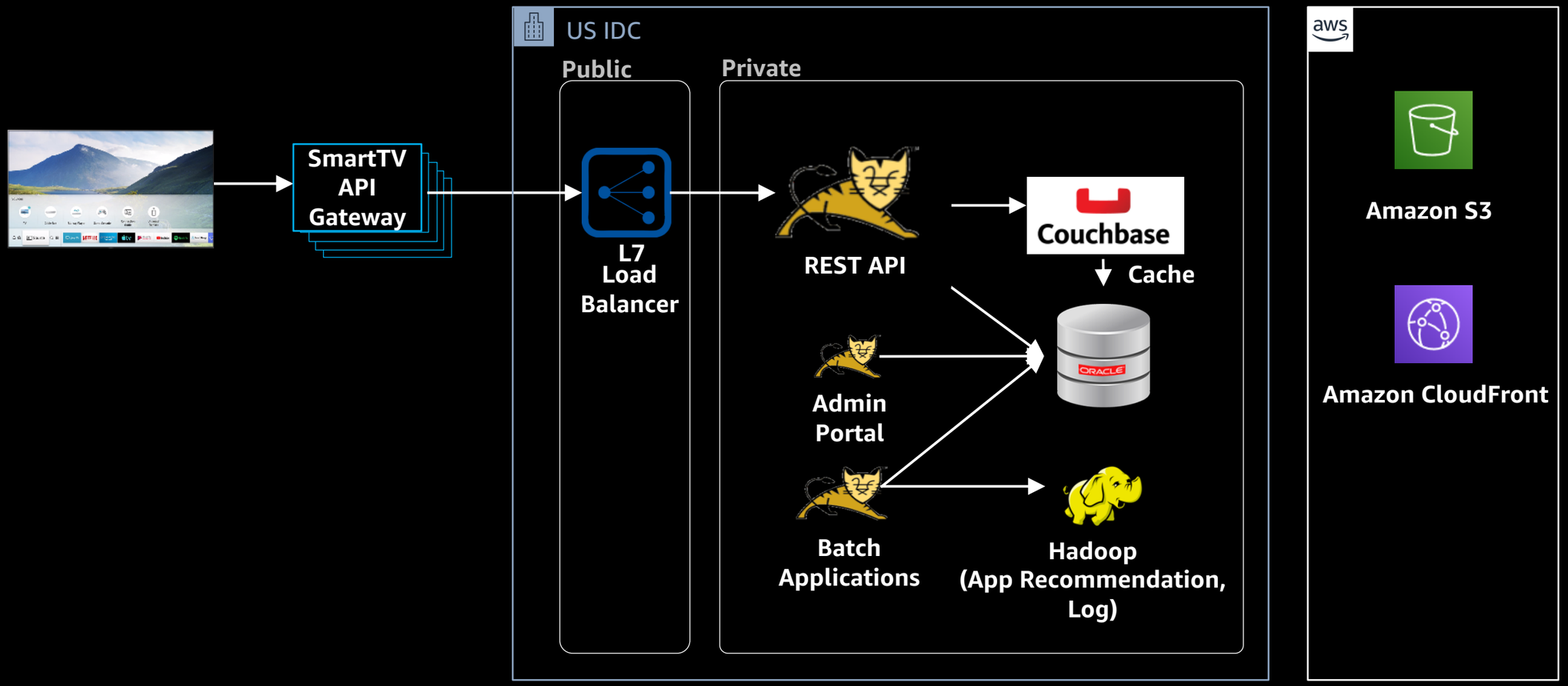
Steps

LONG AWS MIGRATION JOURNEY

- AWS migration based on cloud-native architecture and microservices
  - Re-architecting applications into Microservices
  - Migration is still ongoing after 4 years (including Architecture enhancements)
- Building microservices one by one
  - APIs with less blast radius first
  - Since 2015, all the REST APIs have been running behind the API gateway
  - We could segregate applications without any changes required from SmartTV client
- Oracle to MySQL Amazon Aurora with multi-region
- Managed service
  - Application and architecture optimization is required

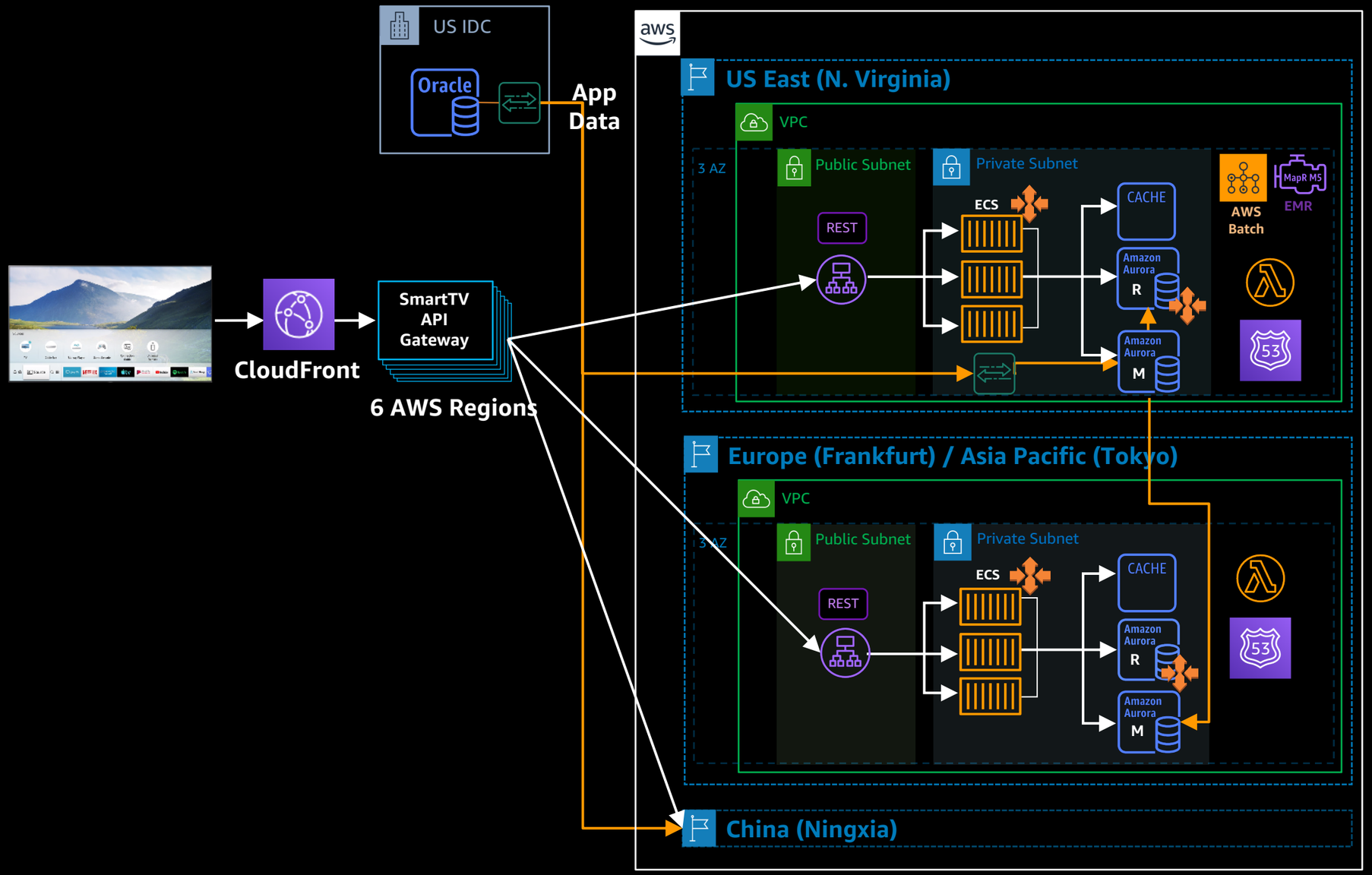
Architecture

BEFORE



Architecture

AFTER



- API Gateway in 6 regions
- Apps application in 4 regions
- Auto Scaling with Amazon ECS and Amazon Aurora
- Region aware configurations are managed by Amazon Route 53 and Spring Cloud Config

AWS ECS and AWS Fargate



AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Containerization

BENEFIT

- Application mobility is enhanced
- Scaling is easier compared to VM or Amazon EC2
- Resource utilization can be optimized
- Cost can be heavily optimizable

What we did to Containerizing

SUMMARY OF CONTAINER FROM MY END

1. Change the way of thinking
   - It’s not easy to change your dev principles and also your stateless application
   - If your servers can be terminated at anytime, how should we manage it?
   - Unable to use local volume and have to save into shared storage
2. Build once, Deploy anywhere
   - Enabling the ECR Immutable option – pull image from Staging ECR and push to Production ECR without build
3. Externalized configuration using Spring Cloud Config
   - All application configuration files are changed from local to Spring Cloud Config (*AWS region aware configuration)
   - YAML files are saved into S3 bucket
   - “java -jar apps-xxx-api.jar –Denv={env}” only in alpine image

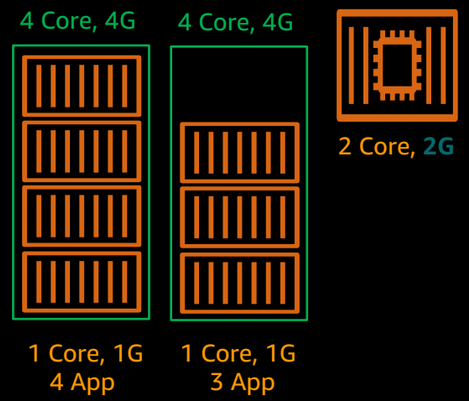
Migration phases



Container migration step

CONTAINER PHASE 1 – AMAZON ECS

- **Amazon ECS**



- Amazon EC2 needed to be created, but are not created automatically
- Cluster management with AWS Lambda based on resource reservation is needed
  - Container scaling – CPU utilization
  - Amazon EC2 cluster scaling – memory reservation

EC2 ECS vs Fargate

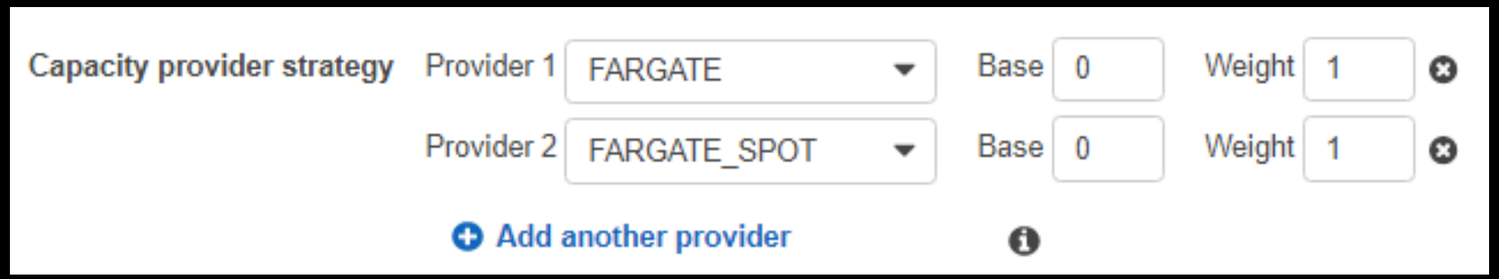
CONTAINER PHASE 2 – COMPARE AMAZON EC2 AMAZON ECS WITH AMAZON ECS AMAZON FARGATE

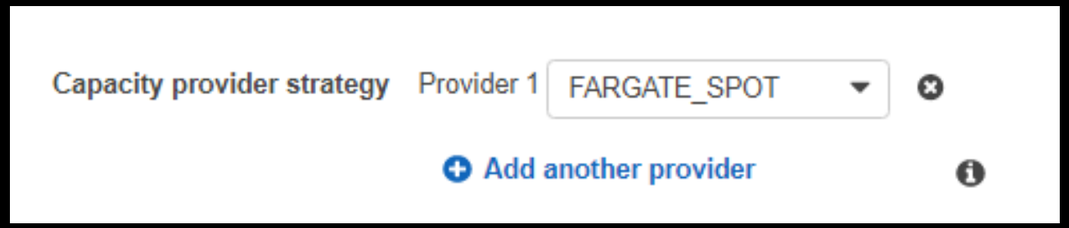
	Amazon ECS	AWS Fargate
Pros	Can choose from various types of Amazon EC2 instances	No need for cluster management Pay for what you use Scaling speed Security
Cons	Amazon EC2 cluster management effort is needed Dynamic port mapping is complicated (32768~61000)	Can’t connect into the Container when application performance issue occurs Limitations ~~• Elastic File System(EFS)~~ **[No More]** • Privileged Container / FPGA

Capacity Provider

CONTAINER PHASE 2 – LAUNCHED ON DEC 3, 2019

- Mix of Fargate and Fargate Spot as capacity providers



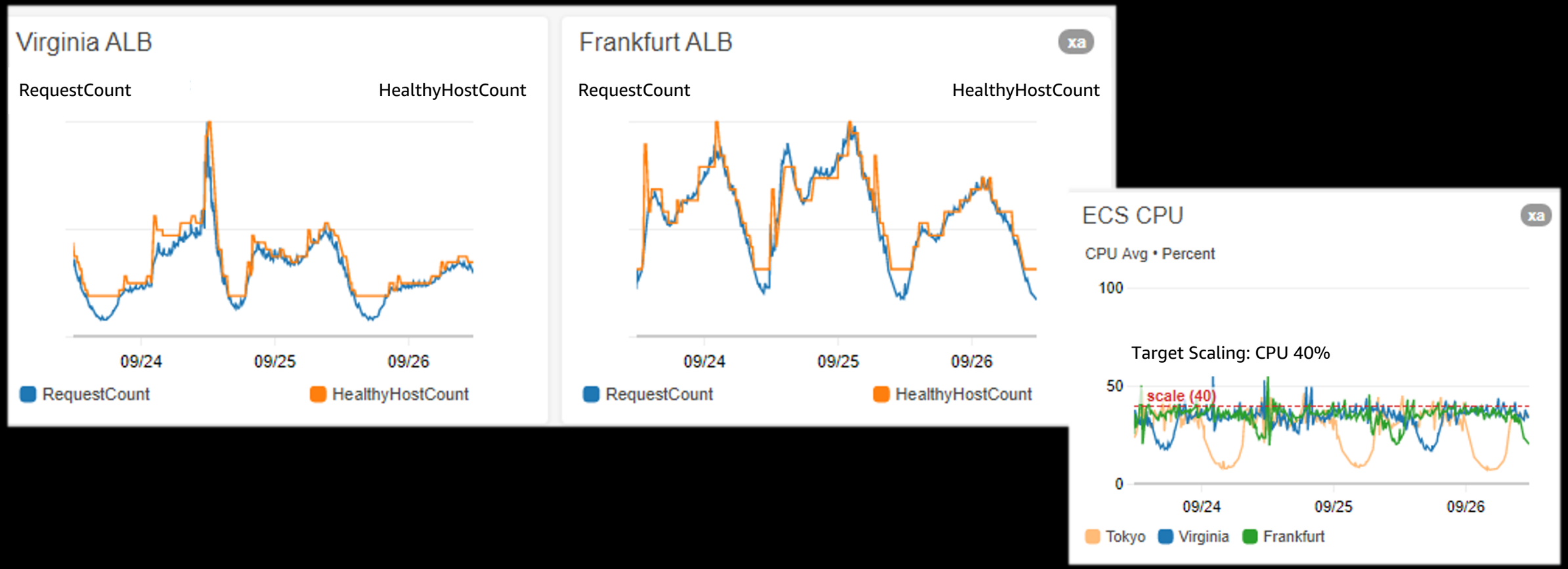


- Base vs. weight

Scaling

BEAUTIFUL CHARTS

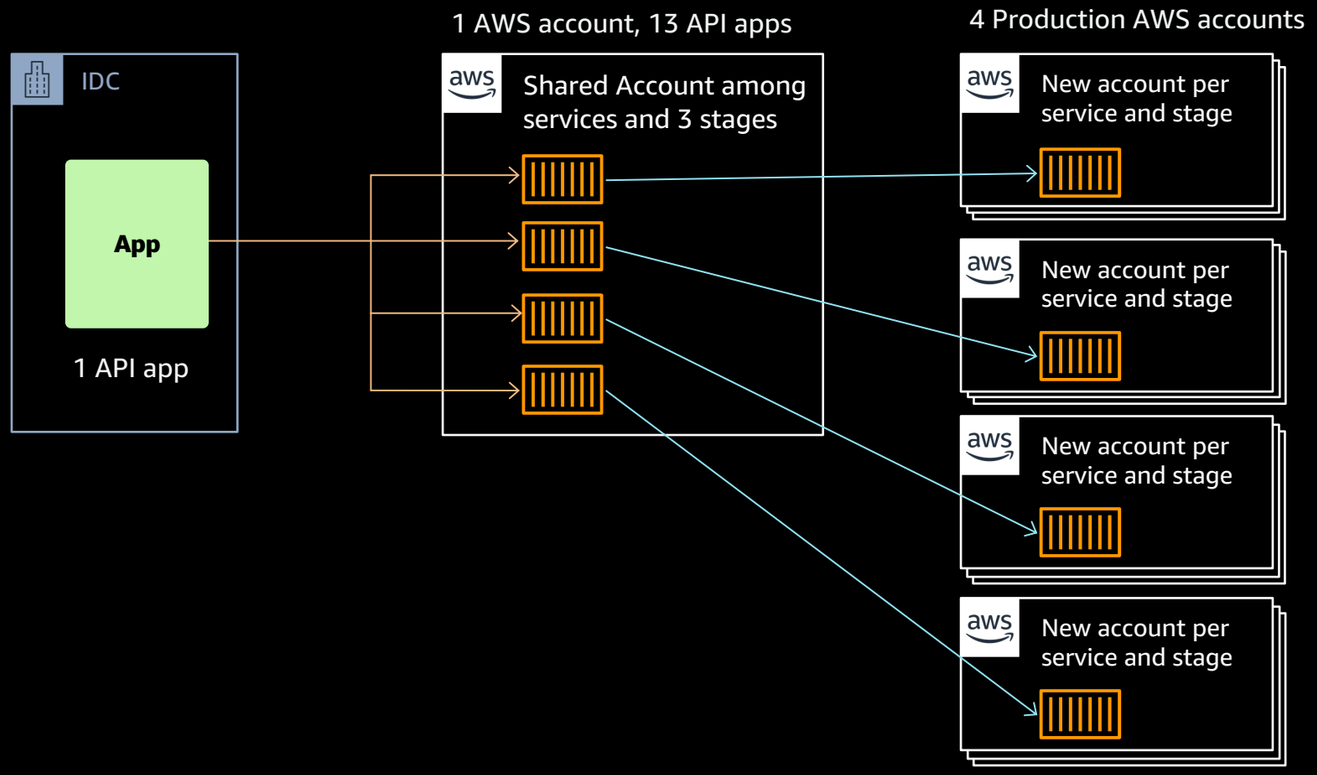
- Step scaling vs. target scaling
- Scaling speed up according to the incoming traffic



Resource utilization in microservices

CPU AND MEMORY RESERVATION

- One application → multiple micro applications (13 applications)
- One IDC / Region → multi-region (4 regions)



Scaling unit should be smaller than before in Microservices

- 256 (.25 vCPU) 0.5 GB
- 1024 (1 vCPU) 2 GB
- 2048 (2 vCPU) 4 GB used according to domain logic and traffic volume

AWS CDK in AWS migration and Fargate



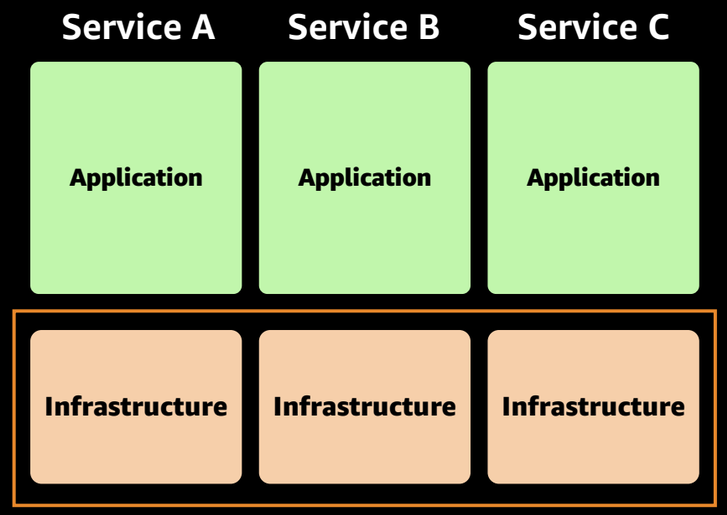
AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

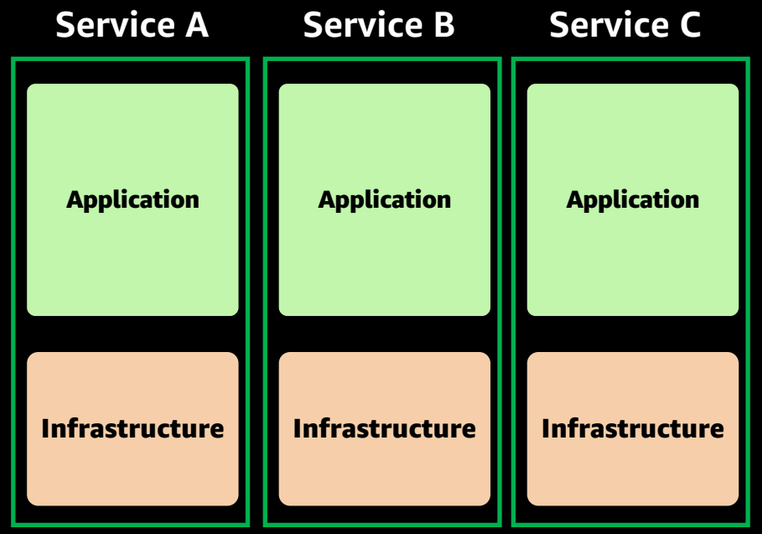
AWS CDK

CHANGES

For better productivity, infrastructure is managed by application developer by using AWS CDK



Shared responsibility where infrastructure by Ops team and application by Service team

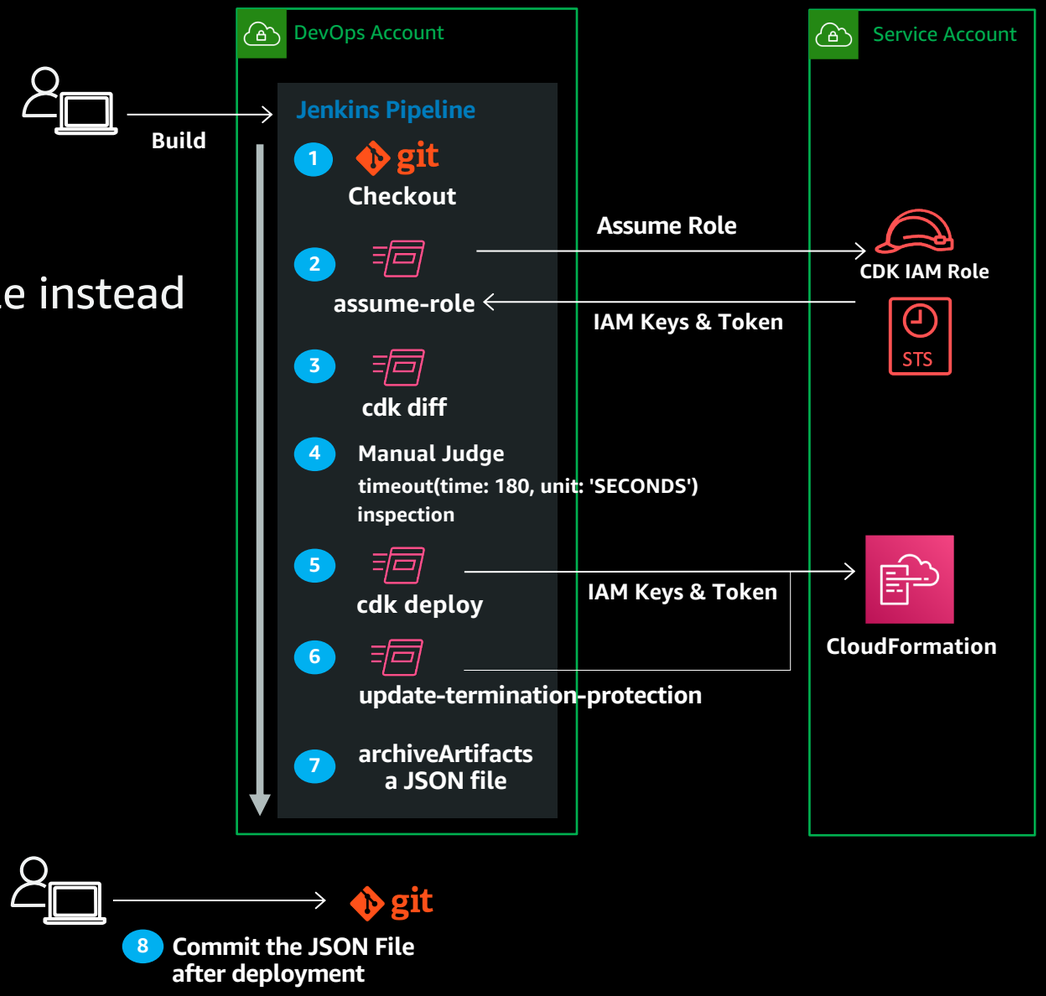


Both application and infrastructure managed by Service team

AWS CDK

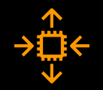
MORE DEVELOPER ORIENTED

- **Security**
  - Developers need more privileges than before
  - No need for individual credential or key; use the assumed role instead
- **History Tracking**
  - CDK is written in code that can be managed in Git
  - Deployment history with Jenkins Pipeline integration
- CDK also has similar characteristics as AWS CloudFormation
- 203 CloudFormation Stacks on production in 4 regions



Shifting in Apps Service

ON PREMISE > VIRTUALIZATION > CONTAINER > SERVERLESS CONTAINER

	On Premise (2017)	Virtualization (2018-2019)	Container (2018-2019)	Serverless (2020)
Computing		 Amazon EMR	 ECS  AWS Batch  Auto Scaling	 Fargate  Lambda  Auto Scaling
Network and contents delivery				
Storage	 CACHE			
Database	 Oracle	 Aurora in 4 regions  Auto Scaling		 Aurora Serverless
Monitoring	APM	CloudWatch and X-Ray		
Security and automation		  IAM role only  Parameter Store		 CDK

Zero-VM

APM: App Performance Monitoring

Cost Optimization

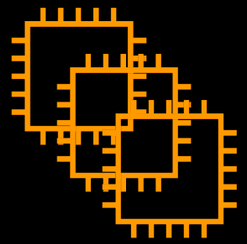
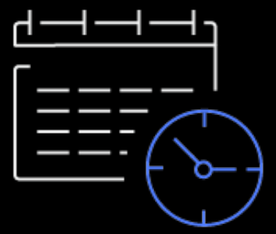
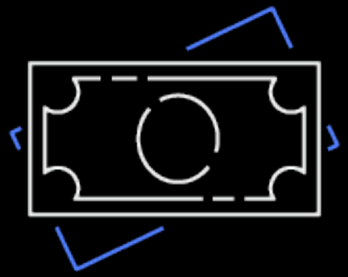


© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Cost Optimization

HOW TO COMPUTE RESOURCE COST

- The Components of Compute Cost

(total compute cost) = (# of compute units) x (time in use) x (cost per unit)

Cost Optimization

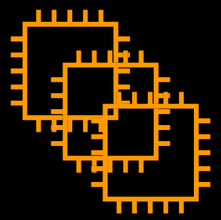
WITH ECS FARGATE SERVICE

- The Components of Compute Cost



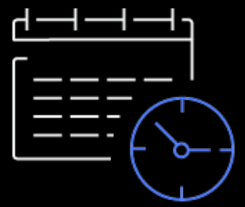
(total compute cost)

=



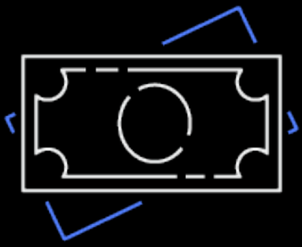
(# of compute units)

x



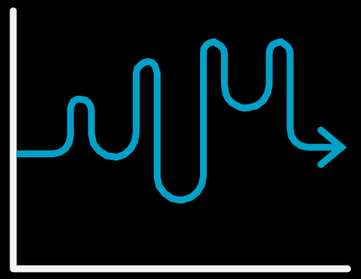
(time in use)

x



(cost per unit)

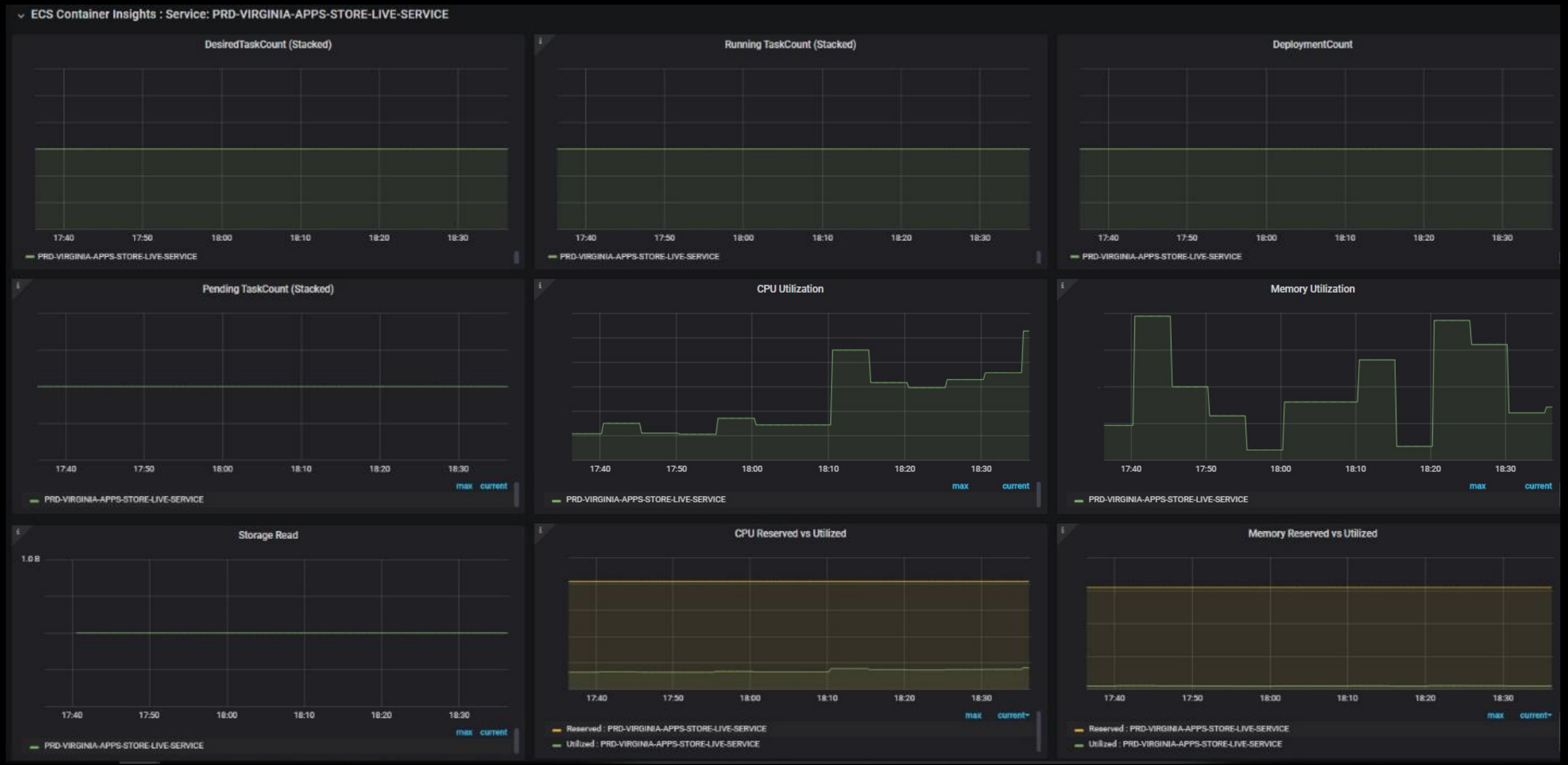
- Fargate Pay for containers per-second with no long-term commitment



Resource capacity needs can change rapidly

Cost Optimization

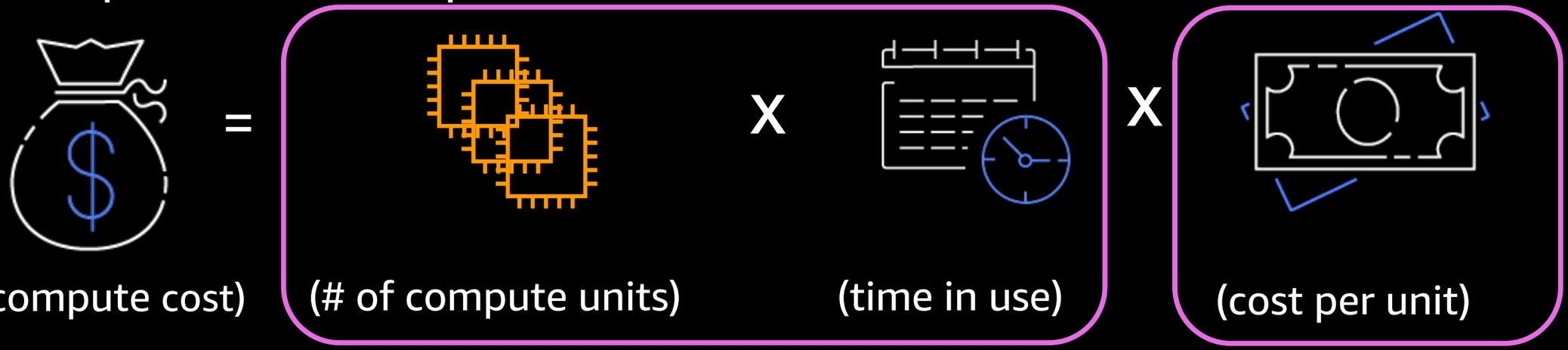
CONTAINER MONITORING SYSTEM



Cost Optimization

HOW TO COMPUTE RESOURCE COST

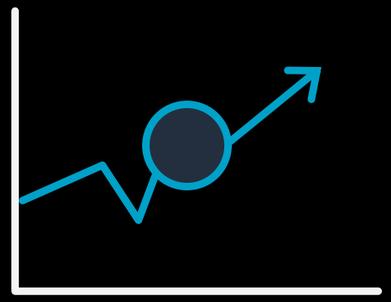
- The Components of Compute Cost



Cost Optimization

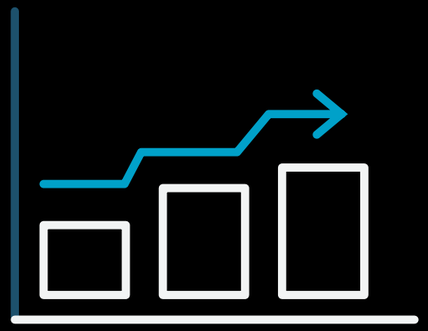
FARGATE SPOT AND SAVINGS PLANS

- Fargate Spot Instances



Spare capacity with **savings of up to 70%** off Fargate standard prices
Fault-tolerant, flexible, stateless workloads

- Compute Savings Plans



Make a 1 or 3-year commitment and receive a significant discount
Baseline compute needs known in advance

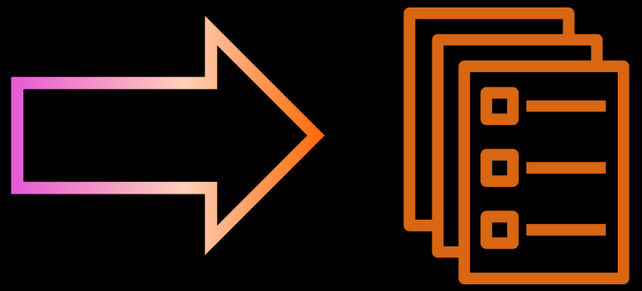
Cost Optimization

COST SAVING

Traditional Server

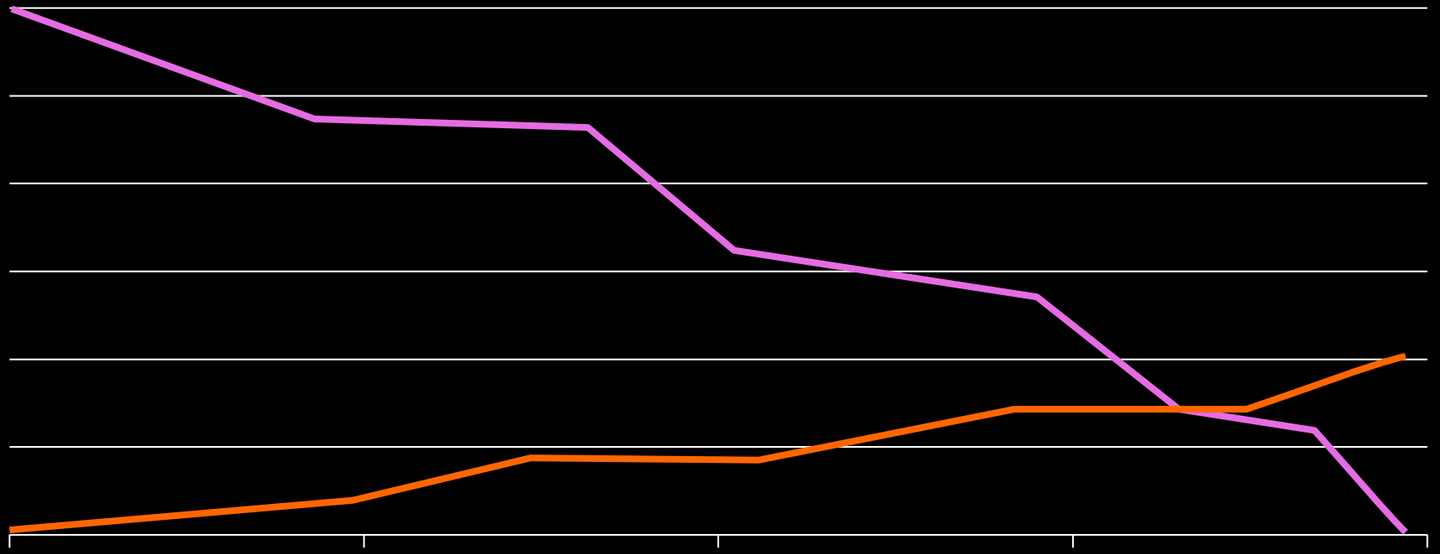


AWS Serverless





63%



Summary

THANKS

Finally, we don't need to do anything after production deployment

Auto Scaling is working fine every day

Nowadays, I’m in a deep sleep every day without any alarm or system failure

Thank you Fargate and Aurora

Thank you!



AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.



Please complete the session survey

AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.