Serverless Architectures

Why is Everyone Moving to Serverless?

Ray Chang

Principal Solutions Architect
Amazon Web Services

Nick Ragusa

Principal Solutions Architect
Amazon Web Services

Agenda

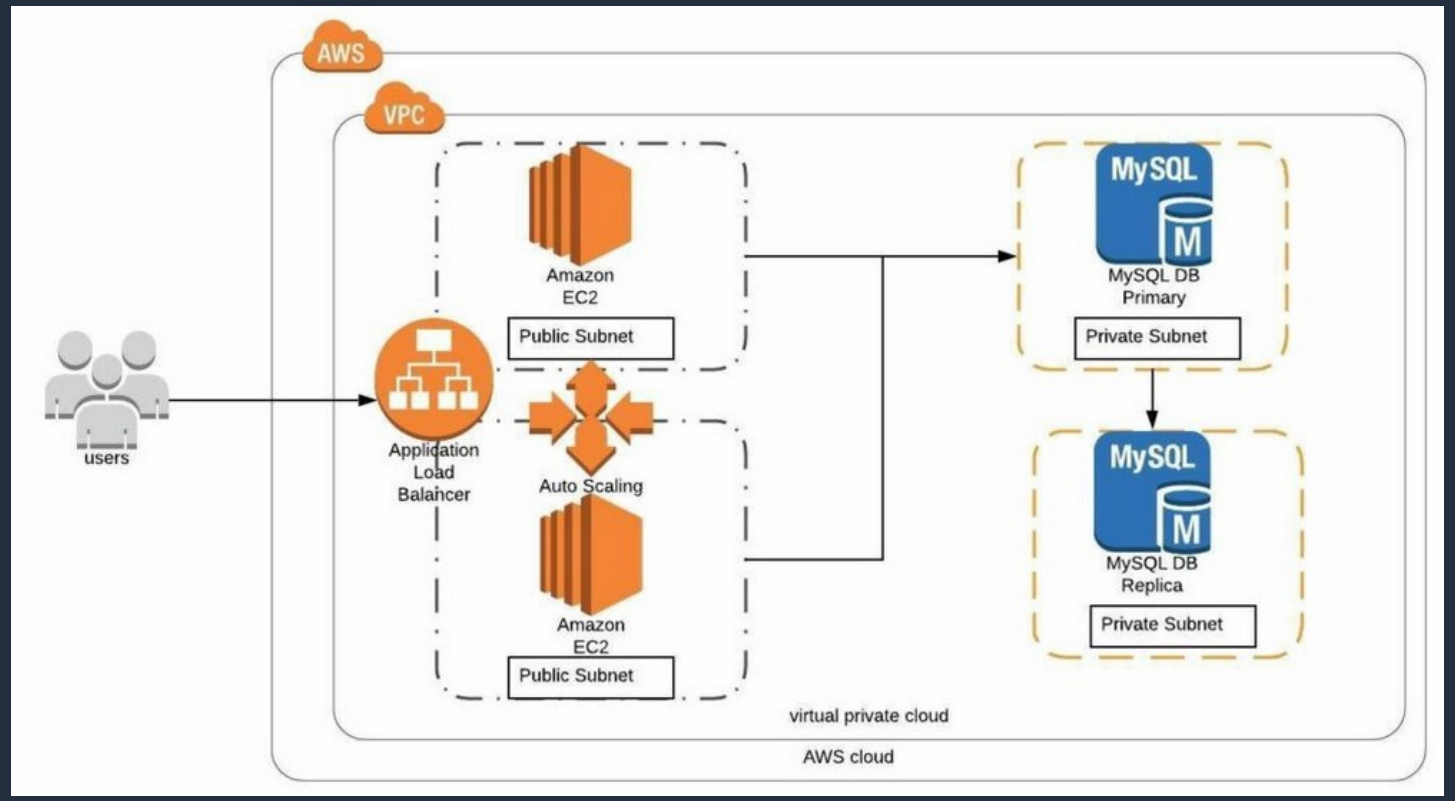
- Where we have come from – servers
- Where to start with Serverless
- Sample Serverless Architecture

Servers

How do we use servers?

- State management
- Monolithic container for functionality
- One version, one server
- Server is an atomic unit of thinking
- The challenges of this model

In the good old days there was one way (EC2)



Today there are more choices!

AWS App Runner
Build and run production web applications at scale

Batch
Fully managed batch processing at any scale

EC2
Virtual Servers in the Cloud

EC2 Image Builder
A managed service to automate build, customize and deploy OS images

Elastic Beanstalk
Run and Manage Web Apps

Lambda
Run Code without Thinking about Servers

Lightsail
Launch and Manage Virtual Private Servers

AWS Outposts
Run AWS Services On Premises

Serverless Application Repository
Assemble, deploy, and share serverless applications within teams or publicly

Elastic Container Registry
Fully-managed Docker container registry : Share and deploy container software, publicly or privately

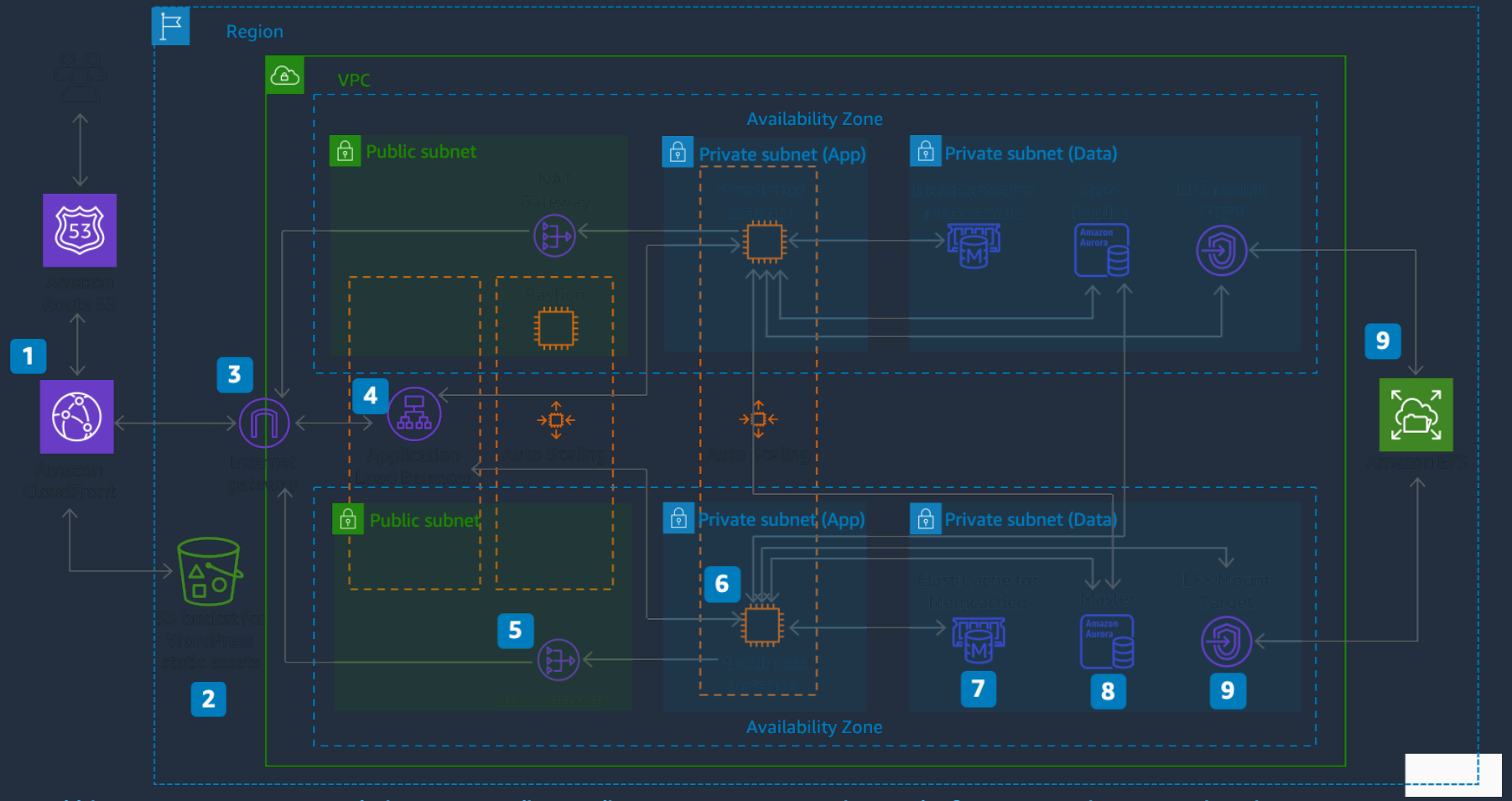
Elastic Container Service
Highly secure, reliable, and scalable way to run containers

Elastic Kubernetes Service
The most trusted way to start, run, and scale Kubernetes

Red Hat OpenShift Service on AWS
Fully managed Red Hat OpenShift service on AWS

Customers love that they can pick the right tool for the job but that comes with some decision fatigue

Example: WordPress hosting on Amazon Web Services



https://docs.aws.amazon.com/whitepapers/latest/best-practices-wordpress/reference-architecture.html

Where to start with serverless

General approach to thinking serverlessly

Features first

Avoid monolithic thinking

Focus on events

Events are triggers that cause action

Statelessness

The key to scaling effectively

Data flow

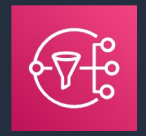
Make data decisions early on

Use the services

Don’t reinvent the wheel

What are serverless services?


Amazon EventBridge


Amazon Simple Notification Service

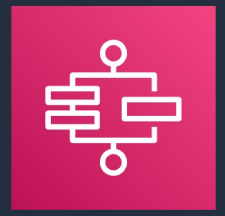

Amazon Simple Queue Service


Amazon API Gateway


Amazon DynamoDB


AWS Lambda


Amazon S3

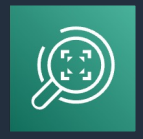

AWS Step Functions


Amazon Kinesis


AWS IoT Core


Amazon Elastic Transcoder

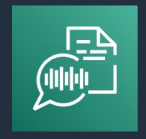
Integrating with other AWS services



Amazon Rekognition

Amazon Comprehend

Amazon Textract



Amazon Transcribe

Amazon Translate

How AWS Lambda fits in

Attributes

- Runs on demand
- Supports many runtimes
- Responds to events
- Stateless
- Automatically scales

Best practices

- Avoid lifting-and-shifting
- 1 Lambda function per purpose
- Keep functions small
- Choose the right runtime
- Use functions for business logic and plumbing between services
- Include security

Good serverless practices

- Infrastructure is disposable
- Asynchronous versus synchronous processing
- You can mix and match runtime
- Security still top priority
- Automation
  - AWS Serverless Application Model (AWS SAM)
  - Serverless framework

Introducing AWS SAM

AWS Serverless Application Model (AWS SAM)

- AWS CloudFormation extension optimized for serverless
- Serverless resource types: Functions, APIs, tables
- Supports anything CloudFormation supports
- Open specification (Apache 2.0)

To learn more, visit:
https://aws.amazon.com/serverless/sam/

Example AWS SAM template

```yaml
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31

Parameters:
  TargetLanguage:
    Type: String
    Default: 'fr es it'

Resources:
  TranslationBucket:
    Type: AWS::S3::Bucket

  TranslatorFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: translatorFunction/
      Handler: app.handler
      Runtime: nodejs14.x
      MemorySize: 128
      Environment:
        Variables:
          targetLanguage: !Ref TargetLanguage
      Policies:
        - S3CrudPolicy:
            BucketName: !Ref TranslationBucketName
      Events:
        FileUpload:
          Type: S3
          Properties:
            Bucket: !Ref TranslationBucket
            Events: s3:ObjectCreated:*
            Filter:
```

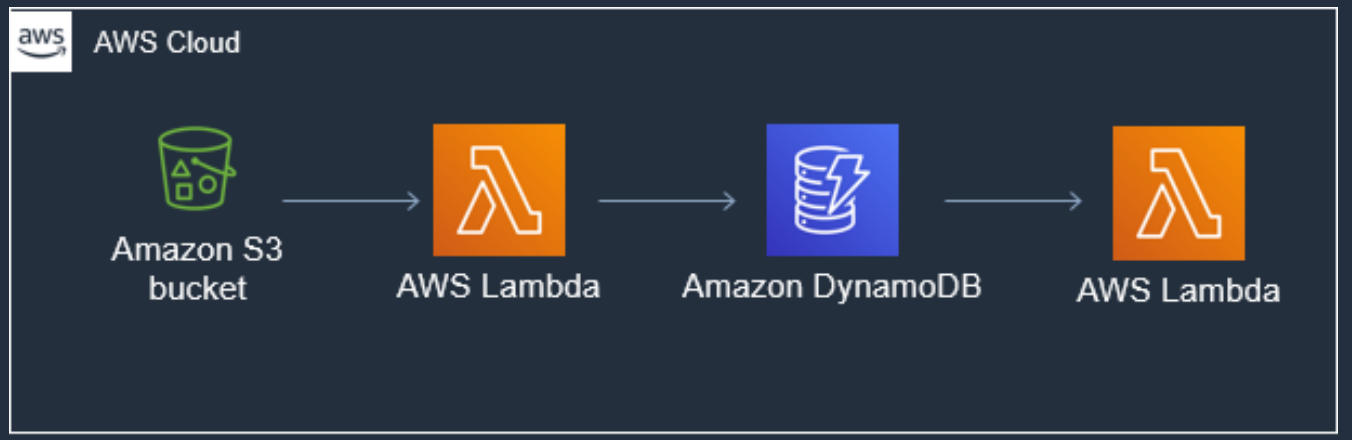
Transforms YAML into infrastructure

```yaml
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31

Parameters:
  TargetLanguage:
    Type: String
    Default: 'fr es it'

Resources:
  TranslationBucket:
    Type: AWS::S3::Bucket

  TranslatorFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: translatorFunction/
      Handler: app.handler
      Runtime: nodejs14.x
      MemorySize: 128
      Environment:
        Variables:
          targetLanguage: !Ref TargetLanguage
      Policies:
        - S3CrudPolicy:
            BucketName: !Ref TranslationBucketName
      Events:
        FileUpload:
          Type: S3
          Properties:
            Bucket: !Ref TranslationBucket
            Events: s3:ObjectCreated:*
            Filter:
              S3Key:
                Rules:
                  - Name: suffix
                    Value: '.txt'
```



Whiteboarding

Sample Scenario: Form upload

Create a serverless application to support a student feedback form submitted from a webpage

- Incoming responses must be translated into English
- Allow user to upload image with a response
- Email any negative comments immediately
- Only allow signed-in students to post feedback

Wrap up

Thank you!

Ray Chang
rchang@amazon.com

Nick Ragusa
ragusan@amazon.com

Please take the session survey:

Track: Application modernization, security, and governance

Session: Serverless Architectures – Why is everyone moving to serverless?