Hands-on Cloud Computing Services

Lezione 3

Gabriele Russo Russo

University of Rome Tor Vergata, Italy

A.Y. 2025/26

TOR VERGATA
UNIVERSITÀ DEGLI STUDI DI ROMA

Beyond AWS CLI

- AWS CLI enables faster interaction compared to the web console
- Commands can be arranged into scripts to solve tasks
- Complex use cases may need a more flexible approach
    - e.g., how to programmatically interact with Cloud resources?

Boto: Python API for AWS

- Boto: AWS SDK for Python
- Enables developers to create, configure, and manage AWS services
- Easy to use, object-oriented API
- Similar APIs available for other languages as well
- We'll use **boto3**:
  `https://boto3.amazonaws.com/v1/documentation/api/latest/index.html`

Configuring boto3

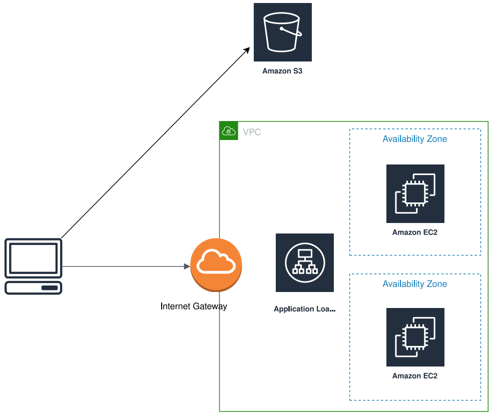
- Boto shares the same configuration of AWS CLI (default region, . . .)
  - If CLI has been configured on your PC, boto3 works out-of-the-box
  - Can be overridden using environment variables or hard-coded settings
- Important issue: providing credentials to Boto
  - Especially important when using boto on remote servers
- Several ways to provide credentials (and configs): `https://boto3.amazonaws.com/v1/documentation/api/latest/guide/credentials.html`

Examples

1. List objects in our bucket: `s3list.py`
2. List EC2 instances: `ec2list.py`

Example: Photogallery

- Extend PhotoGallery with the following features:
  - display pictures stored in a S3 bucket, along with their upload time
  - users can upload pictures



Solution

- Source code: `photogallery_v2`
- How to provide credentials to boto3 to access the bucket?
  - Create a IAM Role for EC2
  - Attach the pre-defined *S3FullAccess* policy
  - Associate the EC2 instance(s) with the new role
- (Check alternative methods in the previous slides)

Cloud Automation

We have introduced a few tools for automation:

- Ansible (with dynamic inventories)
- AWS CLI
- AWS SDK (e.g., boto3)

Enough for infrastructure management?

Infrastructure-as-Code (IaC)

- Define and manage the infrastructure by means of a set of text files, instead of a user interface (CLI, Web, ...)
- Use simple text files to describe your resources (e.g., VMs, security groups, networks)
- Update the files to update the infrastructure
- Benefits:
  - Reduced costs
  - Reduced risks
  - Faster operations
  - Important to enable DevOps practices

Terraform

- Free multi-platform tool for IaC (`www.terraform.io`)
- Can be used with several target platforms (AWS, Azure, VMWare, CloudFlare, ...)
- Infrastructure defined using the *HashiCorp Configuration Language* (HCL)
- Key concepts: Providers + Resources (e.g., “AWS” and “ec2_instance”)

Terraform

- Free multi-platform tool for IaC (`www.terraform.io`)
- Can be used with several target platforms (AWS, Azure, VMWare, CloudFlare, ...)
- Infrastructure defined using the *HashiCorp Configuration Language* (HCL)
- Key concepts: Providers + Resources (e.g., “AWS” and “ec2_instance”)

Terraform is free to use, but not fully open-source since 2023. An open alternative exists: OpenTofu

Terraform + AWS

Requirements:

- AWS CLI installed and configured
- Terraform installed (I am using Terraform 1.10)

We create `example.tf` and run:

- `$ terraform init`
- `$ terraform validate # check syntax`
- `$ terraform apply`
- `$ terraform show`
- `$ terraform apply # nothing to do`

Terraform + AWS

We now **update** `example.tf` adding a tag to the instance:

- Edit `example.tf` adding a tag to the instance
- `$ terraform apply`
- Edit `example.tf` changing the instance type
- `$ terraform apply`
- Let's destroy all the created resources: `terraform destroy`

Terraform: beyond this example

- Resource definitions not limited to EC2 instances!
- Remote storage for `tf.state`
- Versioning Terraform code (e.g., git repo)
- Variables to make code more reusable

AWS: Database Services

AWS provides several database-oriented services. Among them:

- DynamoDB (Key-Value NoSQL tables)
- Aurora (relational DBMS)
- ElastiCache (in-memory databases: Memcached, Redis)
- Neptune (graph database)
- Timestream (for time series)
- RDS (Relational Database Service): easily deploy MariaDB, Aurora, PostgreSQL, …

AWS: Database Services

AWS provides several database-oriented services. Among them:

- DynamoDB (Key-Value NoSQL tables)
- Aurora (relational DBMS)
- ElastiCache (in-memory databases: Memcached, Redis)
- Neptune (graph database)
- Timestream (for time series)
- RDS (Relational Database Service): easily deploy MariaDB, Aurora, PostgreSQL, …

We'll use DynamoDB to store picture metadata in PhotoGallery

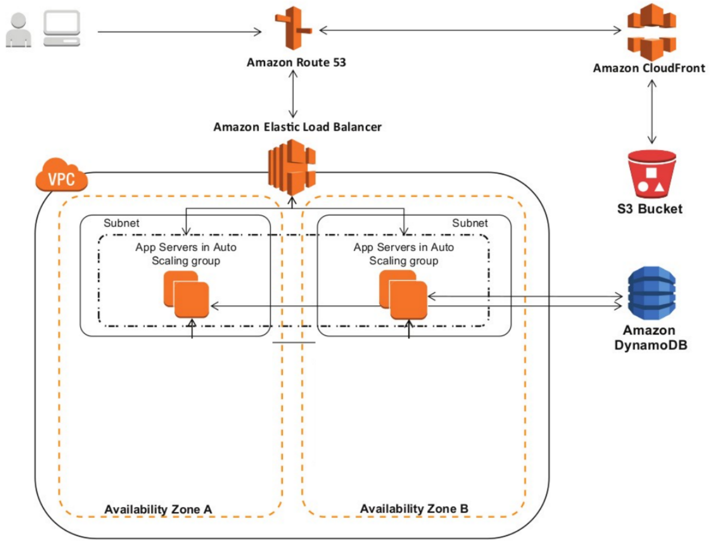
DynamoDB

- Schemaless
- Tables, Items, Attributes
- Primary Key + (optional) Sorting Key
- 2 pricing models:
  - provisioned capacity (default)
  - on-demand
- 2 consistency models:
  - eventual
  - strong

Example: `dynamodb_example/`

Photogallery + DynamoDB

Use DynamoDB to store image tags



Photogallery + DynamoDB: Solution

- Solution: `photogallery_v3`

Simple Queue Service (SQS)

- Fully managed queueing service
- Enables decoupled communication among microservices/components
- Developers can avoid spending effort on a communication middleware
- Standard queues (at-least-once)
- FIFO queues (exactly-once, FIFO order)

- `producer.py` e `consumer.py`

Idea: Photogallery + SQS

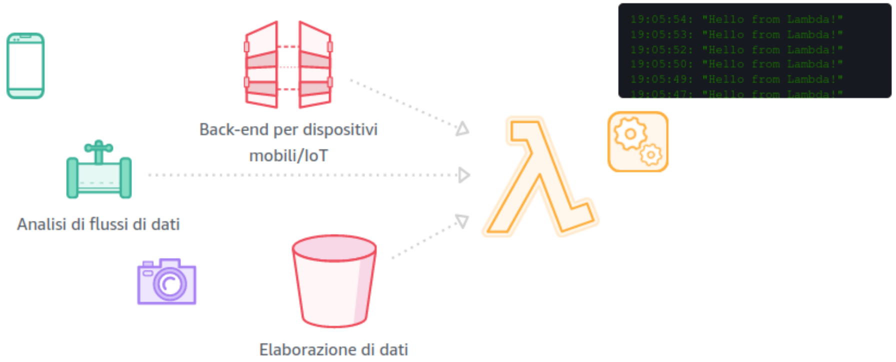
- New images uploaded to S3 in the `pending/` directory
- Image processing (resizing, filters,...) delegated to *workers*
- Web server and workers communicate through SQS (decoupled)

AWS Lambda

- Function-as-a-Service offering by AWS
- Enables the execution of serverless functions
- Functions can be written using many different languages
- Fast scaling from zero to “infinity”
- Fine-grained pay-per-use pricing

Lambda Invocation

Synchronous vs. asynchronous invocation


Back-end per dispositivi mobili/IoT
Analisi di flussi di dati
Elaborazione di dati

```
19:05:54: "Hello from Lambda!"
19:05:53: "Hello from Lambda!"
19:05:52: "Hello from Lambda!"
19:05:50: "Hello from Lambda!"
19:05:49: "Hello from Lambda!"
19:05:47: "Hello from Lambda!"
```

Lambda Invocation (2)

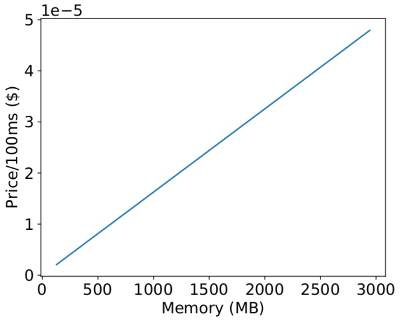
Lambda functions can be invoked in several ways, including:

- AWS CLI
- AWS SDK (boto)
- HTTP(S) endpoints
- automatically in response to events (e.g., new upload to S3)

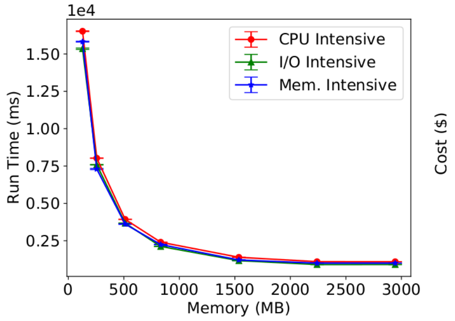
AWS Lambda: Hello World

- Let's create our first Lambda instance
- We can start from a *blueprint*: "Hello, world!"
- We can create **Test** events for our function
- Test the function: observe duration, billed duration, and init duration
- **Cold start**
- We can invoke the function using the SDK and the CLI
  - e.g., `invoke_hello_world.py`

AWS Lambda: Sizing



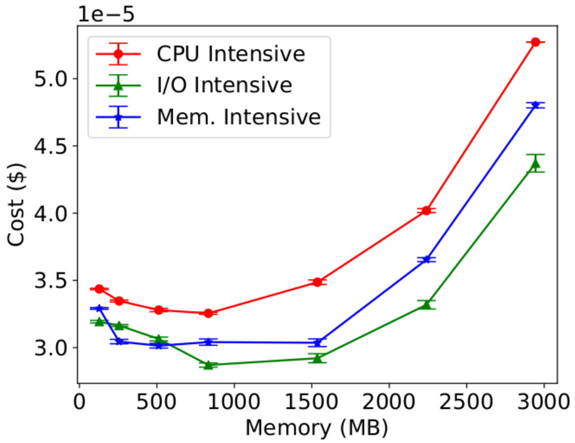
(a) AWS price per 100ms



(b) Run-time of serverless functions on Amazon Lambda

Source: Nabeel Akhtar, Ali Raza, Vatche Ishakian, Ibrahim Matta: **COSE: Configuring Serverless Functions using Statistical Learning**. INFOCOM 2020: 129-138

AWS Lambda: Sizing (2)



(c) Cost for running functions on Amazon Lambda

Invoking Lambda functions via HTTP

- We often want to expose serverless functions as HTTP services (e.g., as part of a larger REST/HTTP API)
- Two possible approaches to do so (each with pros and cons)
- Solution 1: Lambda function URLs
- Solution 2: AWS API Gateway

Function URLs

- New feature launched in 2022
- You can associate an HTTP URL to a Lambda function
- Requests targeted to the URL served by the function
- You can enable it using the web UI or Terraform

Example: Lambda + Function URL

- Toy example: a Pow function
  - Handles GET requests with parameters x and y
  - Returns x^y
- We want a Lambda function to handle HTTP requests
- Source code: `lambda_pow_url`

AWS API Gateway

- AWS Service to create, publish and maintain REST, HTTP and WebSocket APIs
- Developers create APIs integrated with other web or AWS services (EC2, Lambda, . . .).
- Supports authentication mechanisms and fine-grained API management.
- Pricing: ~ 1 \$ per 300M API calls

API Gateway: Key Concepts

- API (e.g., `mydomain.com/app/`)
- Deployment (e.g., test, stage, production)
- Resource (e.g., `/app/posts/`)
- Methods (e.g., `/app/posts/` $\rightarrow$ GET)
- Integration (e.g., Lambda functions, custom HTTP backends)

Example: Lambda + API Gateway

- Let's integrate Lambda and API Gateway
- We want Lambda functions to handle HTTP requests
- Same example: Pow function
- Source code: `lambda_pow`

API Gateway + CORS

- When calling your APIs from a frontend you will likely encounter issues related to CORS (Cross-Origin Resource Sharing)
- `https://docs.aws.amazon.com/apigateway/latest/developerguide/how-to-cors.html`
- When using the Lambda Proxy integration, it suffices to return the correct headers in the Lambda response (as we did for the previous function)
- Example: web frontend for pow

Function URLs vs API Gateway

	Function URL	**API Gateway**
API Type	HTTP	HTTP, WebSocket
Authentication	IAM	IAM, AWS Cognito, API Key
Custom Domain	(through CloudFront)	Yes
Caching	No	Yes
Resp. Timeout	15 min	29 sec
Integrations	Lambda	Lambda, EC2, …
Extra Price	No	Yes

Serverless Image Processing

We need to process user-provided images for our social network.

For each image, we receive a URL in input and we need to:

- detect objects in the image (e.g., for moderation)
- create a thumbnail for fast preview
- create a grayscale version of the image

Serverless Image Processing: Solution 1

- We define a Lambda function to solve the whole problem
- Source code: `lambda_image/`

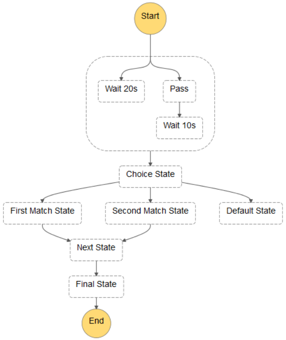
Issues with Solution 1

- Cannot use different languages for different sub-tasks
- Cannot reuse sub-task functionalities (e.g., thumbnail creation)
- Sub-tasks not easily performed in parallel (e.g., grasycale conversion and object detection)

Step Functions

A service for serverless function orchestration.

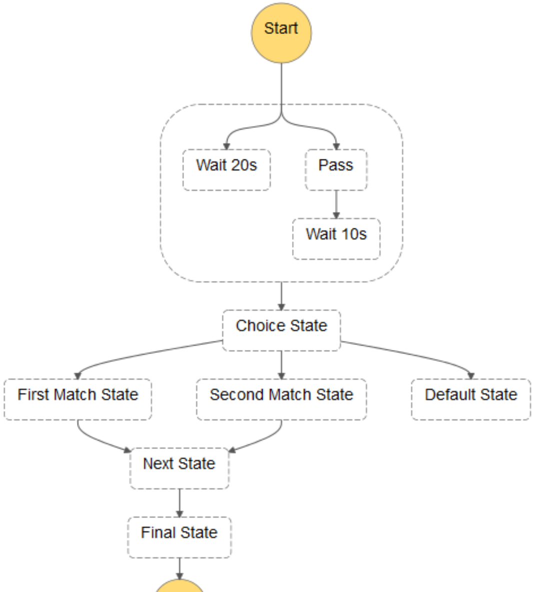
- Workflow definition by means of **state machines**
- Amazon States Language (JSON)
- Pricing: $0.025 per 1,000 state transitions



Step Functions (2)

Various types of state:

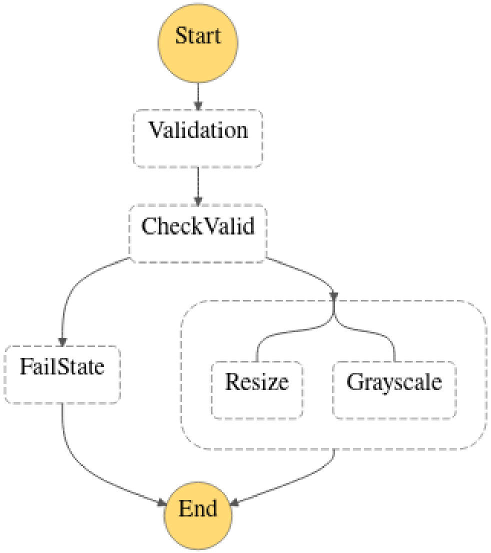
- Task (executes a Lambda function, or a different *activity*)
- Choice
- Fail
- Wait
- Parallel
- Map



Serverless Image Processing: Solution 2

- We rely on 3 Lambda functions:
  - CheckImage()
  - Resize()
  - Grayscale()
- We exploit Step Functions for orchestration
- Source code: `lambda_image2`

Serverless Image Processing: Solution 2



Lambda + EFS

- Our functions relied on S3 to store the image throughout application execution
- S3 not well suited for low-latency data access
- Elastic File System can provide ephemeral storage to Lambda functions

Other AWS Services

Cognito Authentication, authorization, and user management for web and mobile apps

Route 53 DNS management

Elastic IP Re-assign static public IP addresses to your EC2 instances on-the-fly.

Simple Notification Service (SNS) Scalable, push-based messaging service for application-to-person or application-to-application communication. Amazon SNS enables you to send messages or notifications directly to users with SMS text messages to over 200 countries, mobile push on Apple, Android, and other platforms or email (SMTP).

Other AWS Services (2)

An increasing number of Machine Learning-based services…

- Lex (chatbots)
- Polly (synthetic speech)
- Translate
- Rekognition (object and facial image recognition)
- …

Estimating Costs

- AWS provides a “cost calculator”
- `https://calculator.aws/`