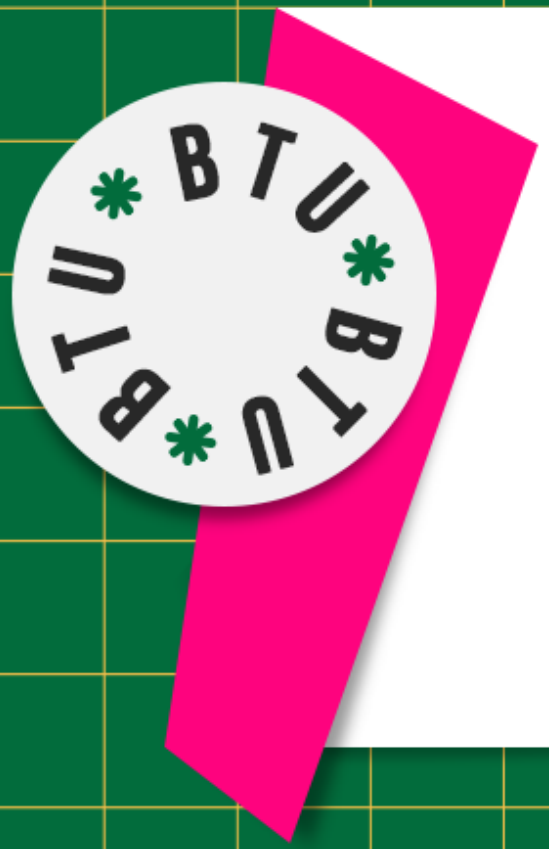


ღრუბლოვანი სისტემების (AWS) ავტომატიზაცია Python-ის საშუალებით

გუჯა ნემსაძე

“Every kid coming out of Harvard, every kid coming out of school now thinks he can be the next Mark Zuckerberg, and with these new technologies like cloud computing, he actually has a shot.”

- Marc Andreessen

საკითხები

- რა არის ღრუბლოვანი სისტემა
- რა საჭიროა ღრუბლოვანი სისტემის ავტომატიზაცია?
- რას შევისწავლით კურსზე
- სამუშაო გარემოს მიმოხილვა
- __Do You Remember Python ??__
- __CLI With Python__

რა არის ღრუბლოვანი სისტემა

Cloud computing is the **on-demand** delivery of compute power, database storage, applications, and other IT resources through a cloud services platform **via the internet** with **pay-as-you-go** pricing.

რა არის ღრუბლოვანი სისტემა

Cloud computing enables you to **stop thinking of your infrastructure as hardware**, and instead **think of it (and use it) as software.**

რა არის ღრუბლოვანი სისტემა

- Hardware solutions are **physical**. This means they require:
  - Space
  - Staff
  - Physical security
  - Planning
  - Capital expenditure
- Guess at theoretical maximum peaks
  - Is there enough resource capacity?
  - Do we have sufficient storage?

What if your needs change?

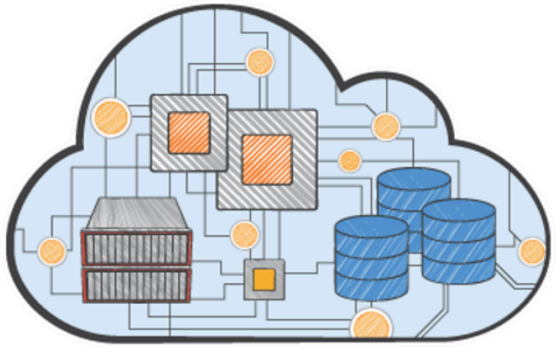
You have to go through the **time, effort, and cost** required to change all these.

რა არის ღრუბლოვანი სისტემა

Software is flexible.

If your needs change, your software can change much more **quickly, easily, and cost-effectively** than your hardware.

რა არის ღრუბლოვანი სისტემა



All-In Cloud

- No upfront investment
- Low ongoing costs
- Focus on innovation
- Flexible capacity
- Speed and agility
- Global reach on demand

On-Premises

- Large initial purchase
- Labor, patches, and upgrade cycles
- Systems administration
- Fixed capacity
- Long procurement cycle and setup
- Limited geographic regions

შევაჯამოთ ღრუბლოვანი სისტემის მინუსები და პლუსები

Pros :

- Disaster Recovery (DR)
- Access your data anywhere
- *Low cost
- Scalability
- *Security
- Benefit from massive economies of scale.
    - Go global in minutes.

Cons :

- Lack of total control
- *Difficult to migrate
- *Requires Internet
- Security and privacy have been an issue
- Cost Concerns

რა საჭიროა ღრუბლოვანი სისტემის ავტომატიზაცია?

Cloud Automation

The Key Fundamental of Cloud Management that you need to practice

რა საჭიროა ღრუბლოვანი სისტემის ავტომატიზაცია?

1. Reduced Costs

Automating your cloud infrastructure reduces the number of human efforts required to manage it. When you implement automation, there are fewer manual errors and the IT team can work on strategic decisions.

2. Faster rate of innovation

Companies can innovate at a faster rate and carry out developments and testing tasks hand in hand by practising automation. How? Because resource scalability and data centre requirements are handled by the cloud provider itself.

3. Extra and better Control

Automation allows organizations to build and implement policies in their cloud footprint right from the start. This gives better control over processes and policies to amend changes whenever required.

4. Process Enhancement

Automation adds intelligence to the system allowing enterprises to simplify IT and business processes thereby streamlining governing policies. This helps IT personnel to spend less time on repetitive tasks and more time on strategic decisions.

5. Implement DevOps practices

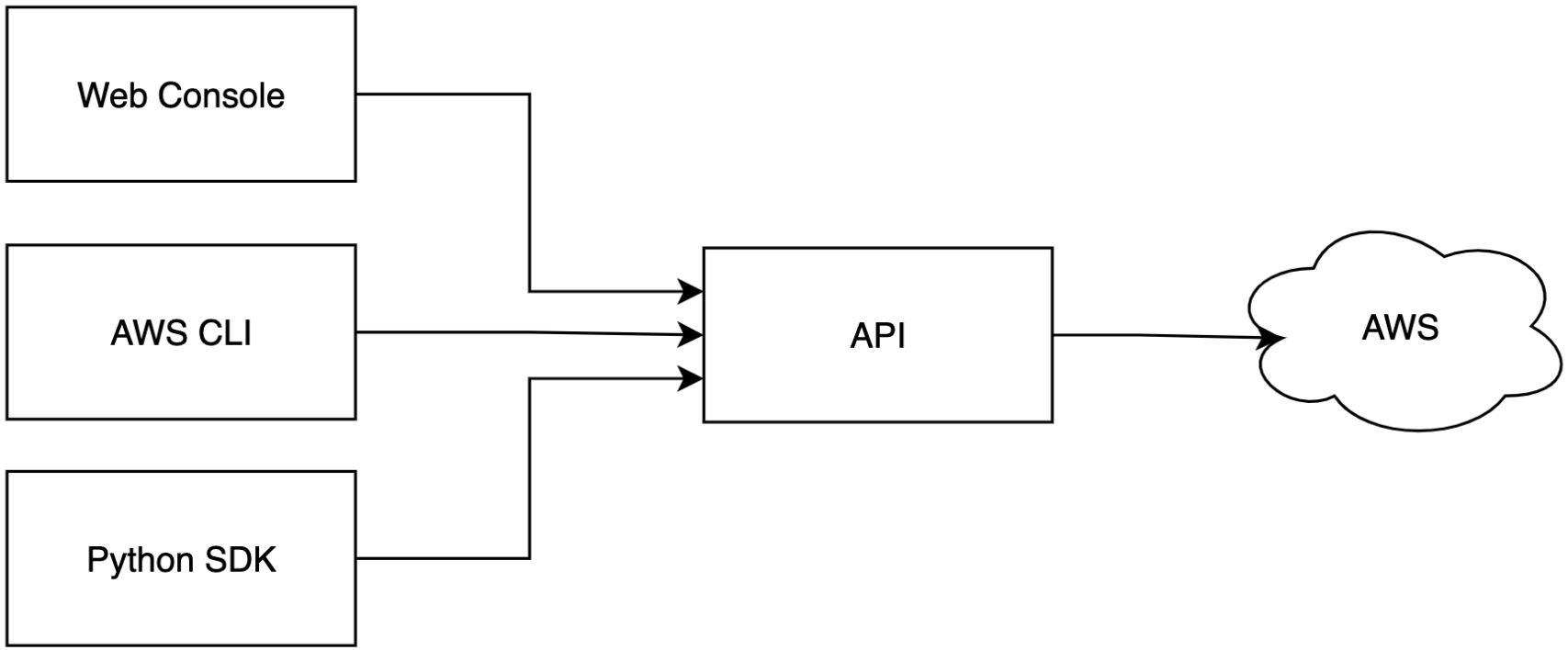
DevOps is a highly popular set of practices that facilitates collaboration between developers and IT operations team within an organization. The key benefits of DevOps include faster processes and stable IT environment. DevOps and practising automation can lead an organization to manage the cloud efficiently with fewer resources.

რას შევისწავლით კურსზე

1. Python-ის დაპრ. ენის გამოყენებით AWS **S3** (საცავების) სერვისის ავტომატიზაცია
2. Python-ის დაპრ. ენის გამოყენებით AWS **VPC** (ინფრასტრუქტურის) სერვისის ავტომატიზაცია
3. Python-ის დაპრ. ენის გამოყენებით AWS **EC2** (ვირტუალური სერვერების) სერვისის ავტომატიზაცია
4. Python-ის დაპრ. ენის გამოყენებით AWS **RDS/DynamoDB** (მონაც. ბაზების) სერვისის ავტომატიზაცია
5. Python-ის დაპრ. ენის გამოყენებით AWS **IAM** (პრივილეგიების მართვის) სერვისის ავტომატიზაცია
6. Python-ის დაპრ. ენის და AWS **Lambda - Serverless** ტექნოლოგიის გამოყენებით ავტომატიზაცია
7. Python-ის დაპრ. ენის გამოყენებით AWS **SQS/SNS** (შეტყობინებების) სერვისების ავტომატიზაცია

სამუშაო გარემოს მიმოხილვა

- მომხმარებელი AWS ის პლატფორმაზე
- AWS CLI(Command Line Interface)
- **SDK**
  - **Python(boto3)**



__Do You Remember Python ??__

- Task 1

დაწერეთ კოდი რომელიც აარჩევს და ამოწერს 9 დან 9999 მდე შუალედში მყოფ ყველა არმსტრონგის რიცხვს. რის შემდეგაც დაწერეთ ეს რიცხვები და მათი ჯამი, *თუმცა ჯამისთვის გამოიყენეთ თქვენი დაწერილი რეკურსიული ფუნქცია.*

არმსტრონგის რიცხვებია:
*9 არის არმსტრონგის რიცხვი რადგან 9 = 9^1

*10 არ არის არმსტრონგის რიცხვი რადგან 10 != 1^2 + 0^2

*153 არის არმსტრონგის რიცხვი რადგან 153 = 1^3+ 5^3 + 3^3 = 1 + 125 + 27 = 153

- Task 2

შექმენით მასივი რომელშიც მომხმარებელი შეიყვანს 3 ნებისმიერ წინადადებას.

data = [ ]

შექმენით მეორე მასივი რომლშიც ჩაწერთ საძიებო სიტყვებს მაგალითად : words = ["moon", "sun"]

შექმენით ფუნქცია რომელსაც პირველ პირობად გადასცემთ data ს ხოლო მეორე პირობად words ს.

მაგალითად: def find_in(data, words):

საბოლოოდ თქვენმა ფუქნციამ უნდა დაადგინოს არის თუ არა საძიებო სიტყვები მომხმარებლის მიერ შემოტანილ წინადადებებში და დაითვალოს ისინი.

Bonus:

* დაადგინეთ შემოიტანა თუ არა მომხარებელმა integer ი (try/except ან isnumeric())

__CLI With Python__

[Python Command-Line Parsing Libraries – Argparse, Docopt, and Click]

გადაკეთეთ **Task 1 CLI** *პროგრამად* **Argparse** *ბიბლიოთეკის გამოყენებით*

სახლში სასურველია მოაწყოთ სამუშაო გარემო:

- შევქმნათ მომხმარებელი AWS ანგარიშზე და მივცეთ საჭირო უფლებები (მოწვევის შემდეგ)
- დავაყენოთ AWS CLI
- დავაყენოთ Python
- დავაყენოთ boto3
- *გავწეროთ უსაფრთხოების პარამეტრები AWS CLI-ის გამოყენებით*
- გავტესტოთ გარემო

ლიტერატურა

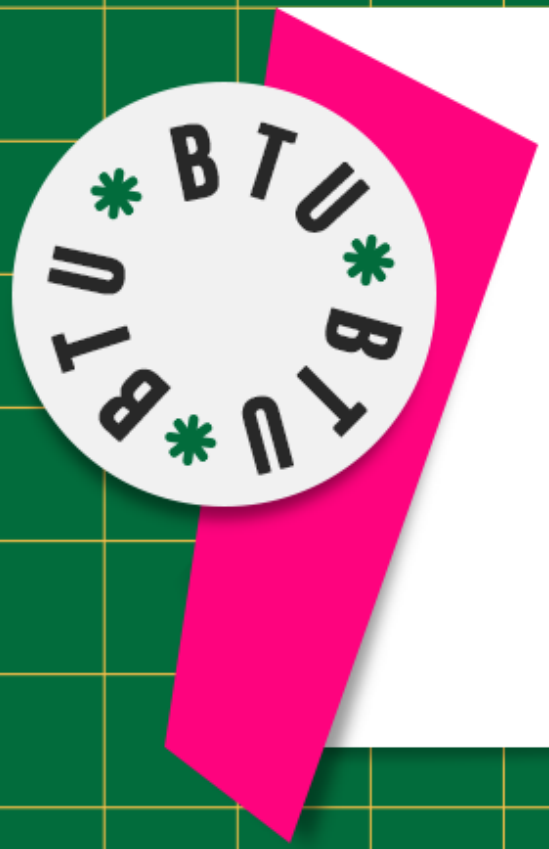
1. AWS Scripted: How to Automate the Deployment of Secure and Resilient Websites with Amazon Web Services VPC, ELB, EC2, RDS, IAM, SES and SNS - კრისტიან ჩერი, 2014წ. , 270 გვერდი
2. Amazon Web Services in Action - ანდრეას ვიტიგი, მიქაელ ვიტიგი, 2016წ., 424 გვ,

მადლობა ყურადღებისთვის!

კონტაქტი

თბილისი 0162, საქართველო
ი.ჭავჭავაძის გამზირი 82
INFO@BTU.EDU.GE
032 2 195 015
WWW.BTU.EDU.GE

გამოგვყევი:



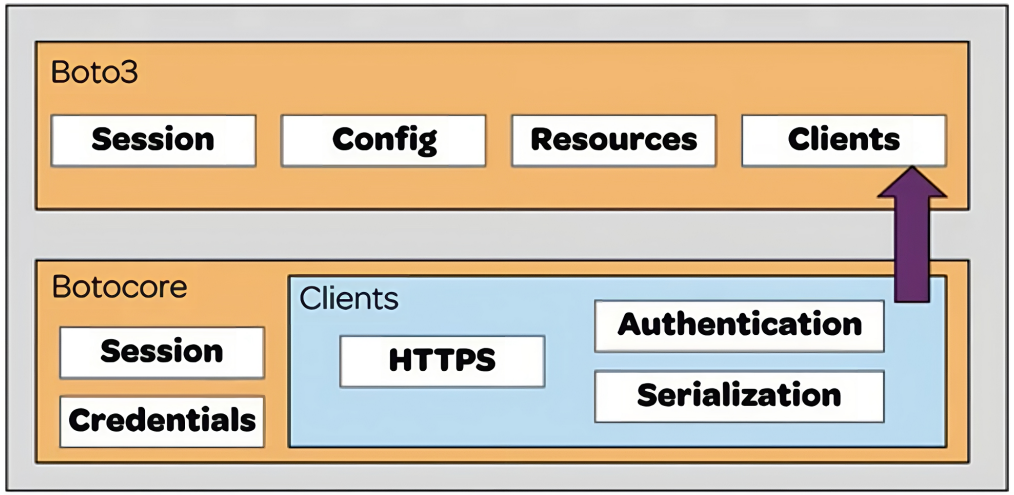
ღრუბლოვანი სისტემების (AWS) ავტომატიზაცია Python-ის საშუალებით

გუჯა ნემსაძე

საკითხები

- AWS SDK for Python (Boto3)
- AWS SDK for Python (Boto3) Prepare Working Env
- AWS SDKs
- AWS CLI && Boto3
- AWS SDK for Python (Boto3) S3 (script)
- Bonus Tasks
- Discussion

AWS SDK for Python (Boto3)



Boto3 is built atop of a library called **Botocore**, which is shared by the **AWS CLI**. Botocore provides the low level clients, session, and credential & configuration data. Boto3 builds on top of Botocore by providing its own session, resources and collections.

Clients

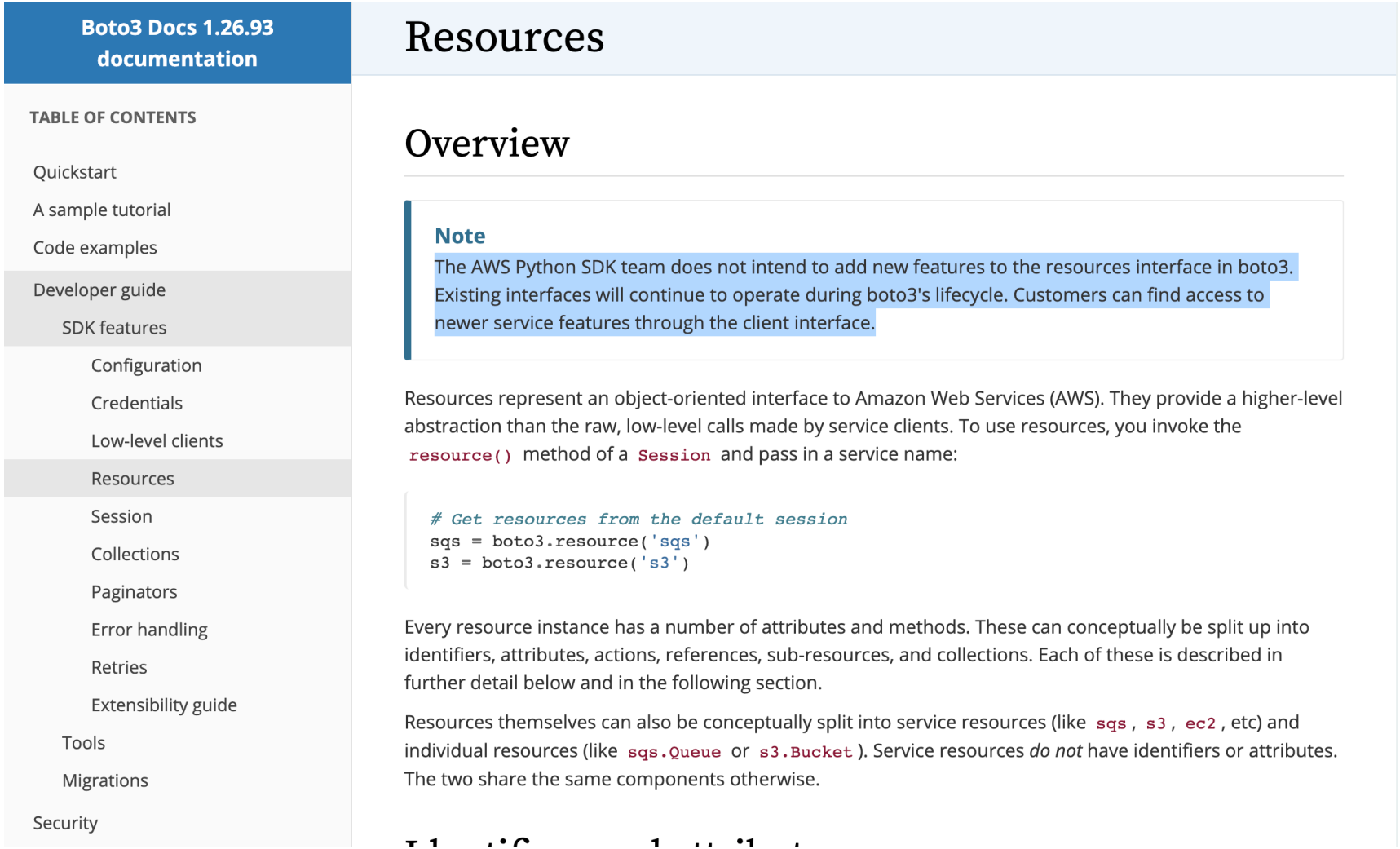
- Exposes Botocore clients to developers
- Generated from JSON **service** descriptions
- Provides low-level service access
- Typically maps 1:1 with the service API
- Method names are snake-cased

`ListBuckets` ⇨ `list_buckets`

Resources

- Generated from JSON **resource** descriptions
- Object-oriented API
- Identifiers and attributes
- Actions
- References
- Sub-resources
- Collections

AWS SDK for Python (Boto3)



Boto3 Docs 1.26.93 documentation

TABLE OF CONTENTS

- Quickstart
- A sample tutorial
- Code examples
- Developer guide
  - SDK features
    - Configuration
    - Credentials
    - Low-level clients
    - Resources
    - Session
    - Collections
    - Paginators
    - Error handling
    - Retries
    - Extensibility guide
  - Tools
  - Migrations
- Security

Resources

Overview

Note

The AWS Python SDK team does not intend to add new features to the resources interface in boto3. Existing interfaces will continue to operate during boto3's lifecycle. Customers can find access to newer service features through the client interface.

Resources represent an object-oriented interface to Amazon Web Services (AWS). They provide a higher-level abstraction than the raw, low-level calls made by service clients. To use resources, you invoke the `resource()` method of a `Session` and pass in a service name:

```
# Get resources from the default session
sqs = boto3.resource('sqs')
s3 = boto3.resource('s3')
```

Every resource instance has a number of attributes and methods. These can conceptually be split up into identifiers, attributes, actions, references, sub-resources, and collections. Each of these is described in further detail below and in the following section.

Resources themselves can also be conceptually split into service resources (like `sqs`, `s3`, `ec2`, etc) and individual resources (like `sqs.Queue` or `s3.Bucket`). Service resources *do not* have identifiers or attributes. The two share the same components otherwise.

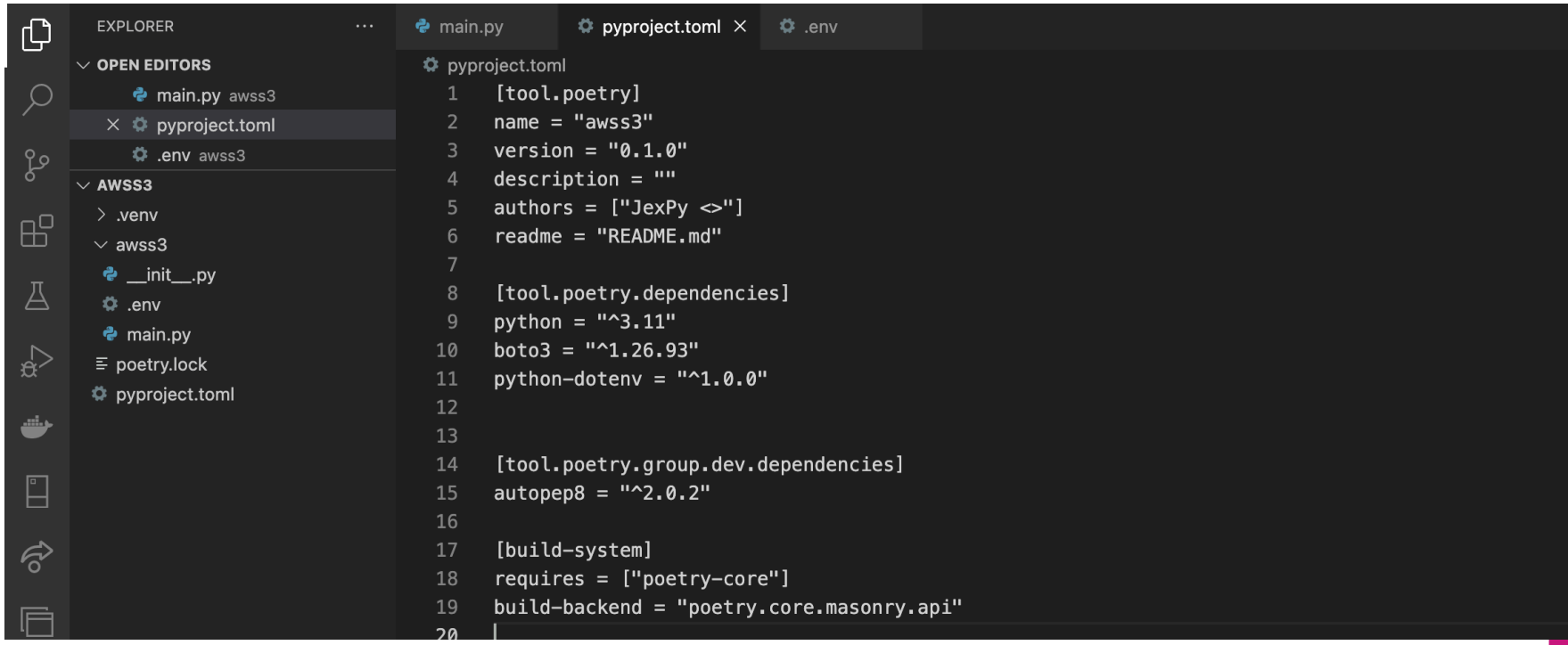
Source: https://boto3.amazonaws.com/v1/documentation/api/latest/guide/resources.html

Discussion / Explanation : https://github.com/boto/boto3/discussions/3563

AWS SDK for Python (Boto3)
Prepare Working Env

1. Install Poetry
2. Run *poetry config --local virtualenvs.in-project true* virtualenvs?
3. Create Project Using Poetry
4. Install Boto3, under created project

Should look like this:

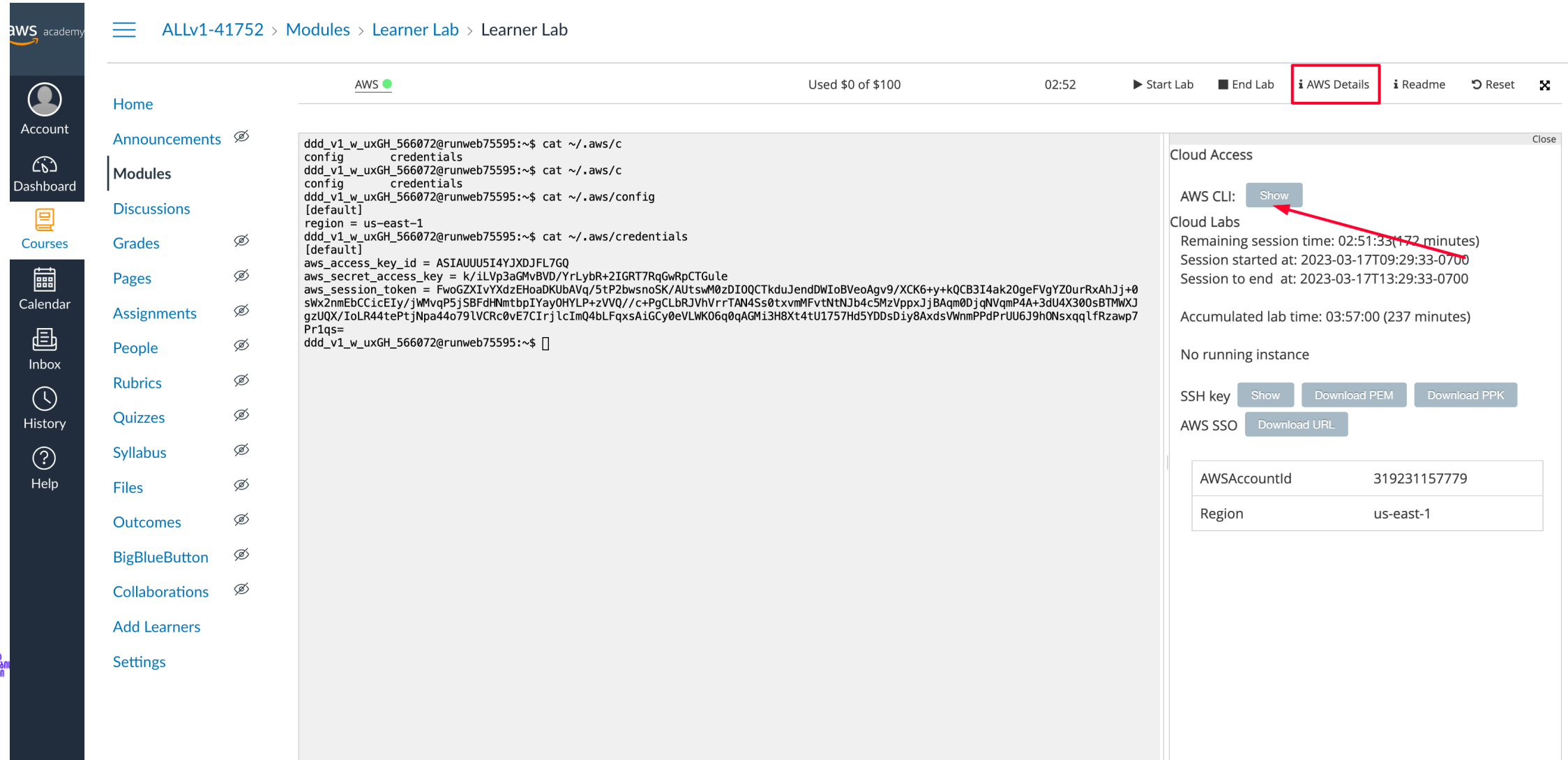


!! Attention !!

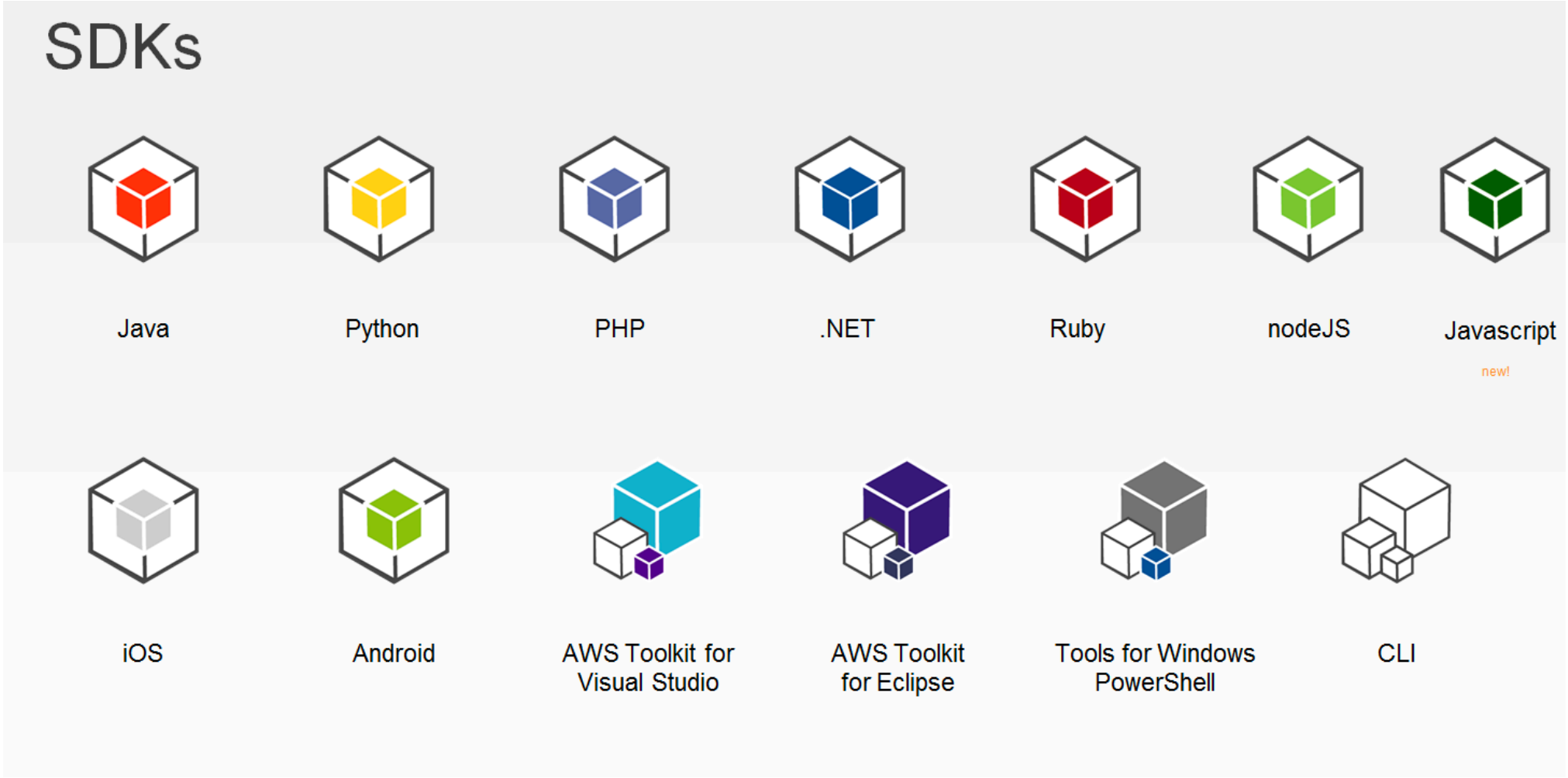
მეგობრებო, გაუფრთხილდით AWSacademy - ის მიერ lab - ზე გამოყოფილ **100$**. მხოლოდ იმ შემთხვევაში ავიღებთ ახალ პროფილებს თუ **100$** სწავლის პროცესში დავხარჯავთ, სხვა ნებისმიერი შემთხვევა შეფასდება როგორც უყურადღებობა, **DevOps ინჟინერია კი პირიქით “პატარა” დეტალების მიმართ უმაღლესი დონის ყურადღებას მოითხოვს.**

AWS SDK for Python (Boto3)
Prepare Working Env

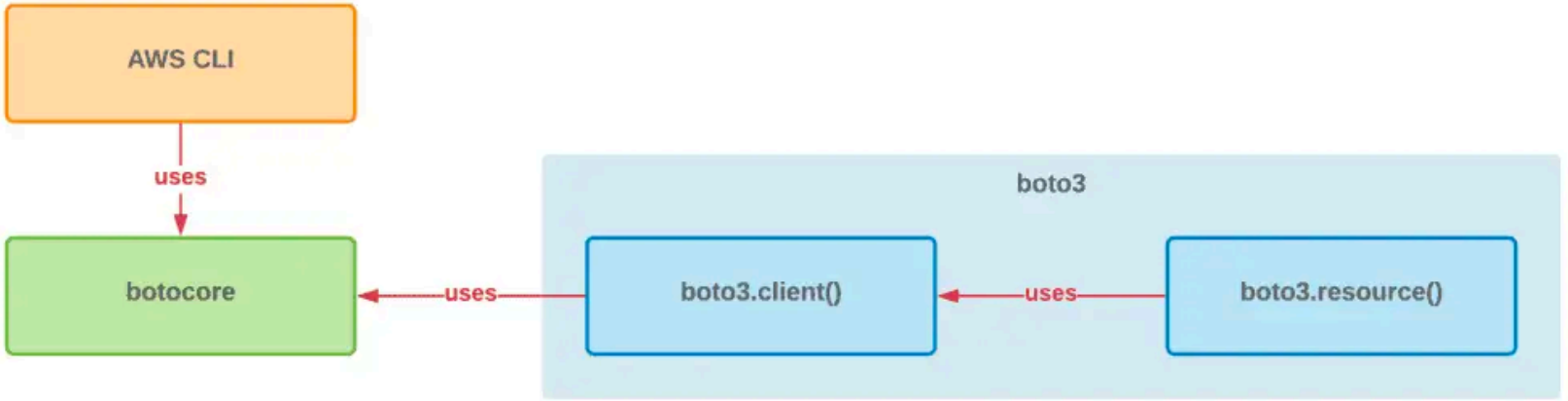
5. Get Credentials



SDKs


SDKs
Java
Python
PHP
.NET
Ruby
nodeJS
Javascript
new!
iOS
Android
AWS Toolkit for Visual Studio
AWS Toolkit for Eclipse
Tools for Windows PowerShell
CLI

AWS **CLI** && **Boto3**



Configure AWS **CLI** (optional)

1. `aws configure --profile pythonBTUautomation`
2. `aws sts get-caller-identity --profile pythonBTUautomation`

```
jexy@Gujas-MBP ~ % aws sts get-caller-identity --profile pythonBTUautomation
{
    "UserId": "AROAUUU5I4YJ7EIUFP5WU:user311515=Elguja_Nemsadze",
    "Account": "319231157779",
    "Arn": "arn:aws:sts::319231157779:assumed-role/voclabs/user311515=Elguja_Nemsadze"
}
```

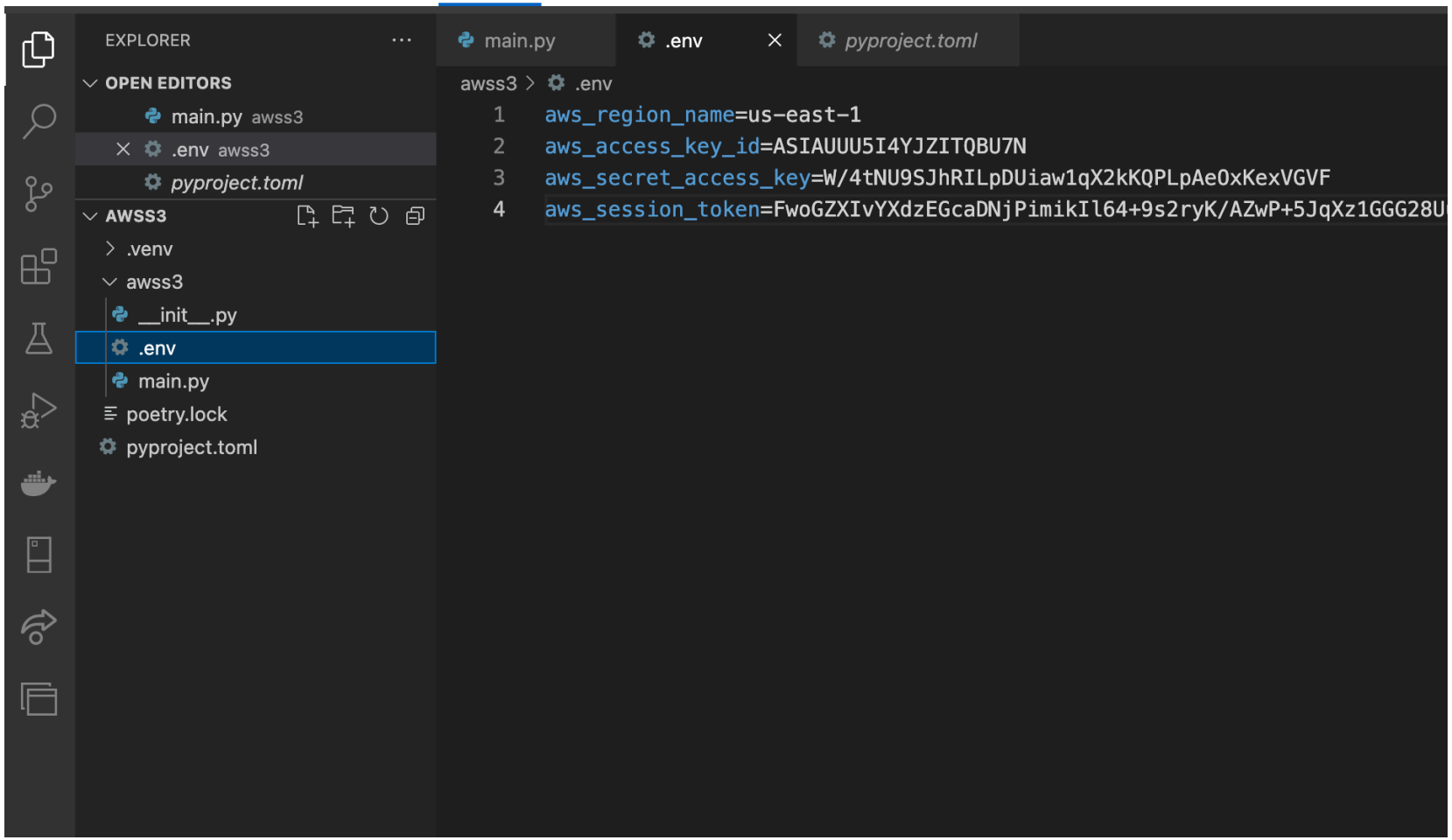
Finished AWS CLI config:

```
jexy@Gujas-MBP ~ % cat ~/.aws/credentials
[default]
aws_access_key_id = [redacted]
aws_secret_access_key = [redacted]
[pythonBTUautomation]
aws_access_key_id = ASIAUUU5I4YJXDJFL7GQ
aws_secret_access_key = k/iLVp3aGMvBVD/YrLybR+2IGRT7RqGwRpCTGule
aws_session_token=FwoGZXIvYXdzEHoaDKUbAVq/5tP2bwsnoSK/AUtswM0zDIOQCTkduJendDWIoBVeoAgv9/XCK6+y+kQCB3I4ak2OgeFVgYZOurRxAhJj+0sWx2nmEbCCicEIy/jWMvqP5jSBFdHNmtbpIYayOH
YLP+zVVQ//c+PgCLbRJVhVrrTAN4Ss0txvmMFvtNtNJb4c5MzVppxJjBAqm0DjqNVqmP4A+3dU4X30OsBTMWXJgzUQX/IoLR44tePtjNpa44o79lVCRc0vE7CIrjlcImQ4bLFqxsAiGCy0eVLWKO6q0qAGMi3H8Xt4tU
1757Hd5YDDsDiy8AxdsVWnmPPdPrUU6J9hONsxqqlfRzawp7Pr1qs=
```

```
jexy@Gujas-MBP ~ % cat ~/.aws/config
[preview]
cloudfront = true
[default]
region = eu-central-1
[profile pythonBTUautomation]
region = us-east-1
output = json
```

AWS SDK for Python (Boto3)
Prepare Working Env

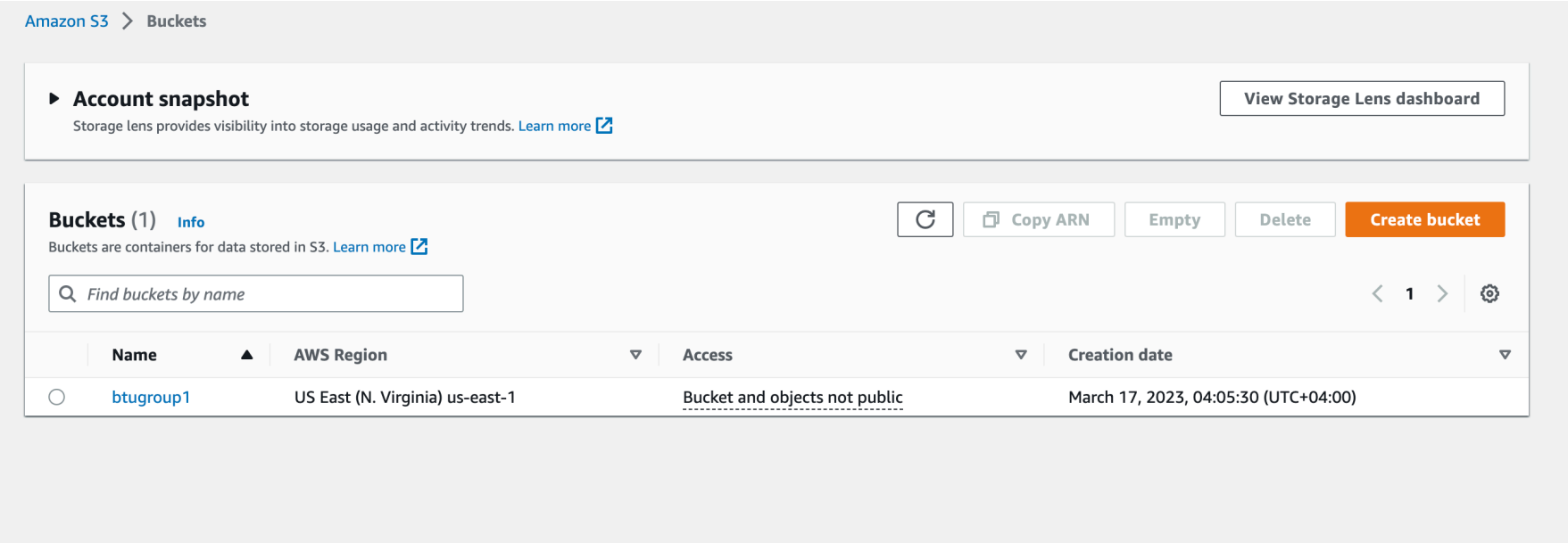
7. Put credentials in .env file



8. Run ***poetry add python-dotenv*** to work with .env files.

AWS SDK for Python (Boto3) S3
Prepare Working Env

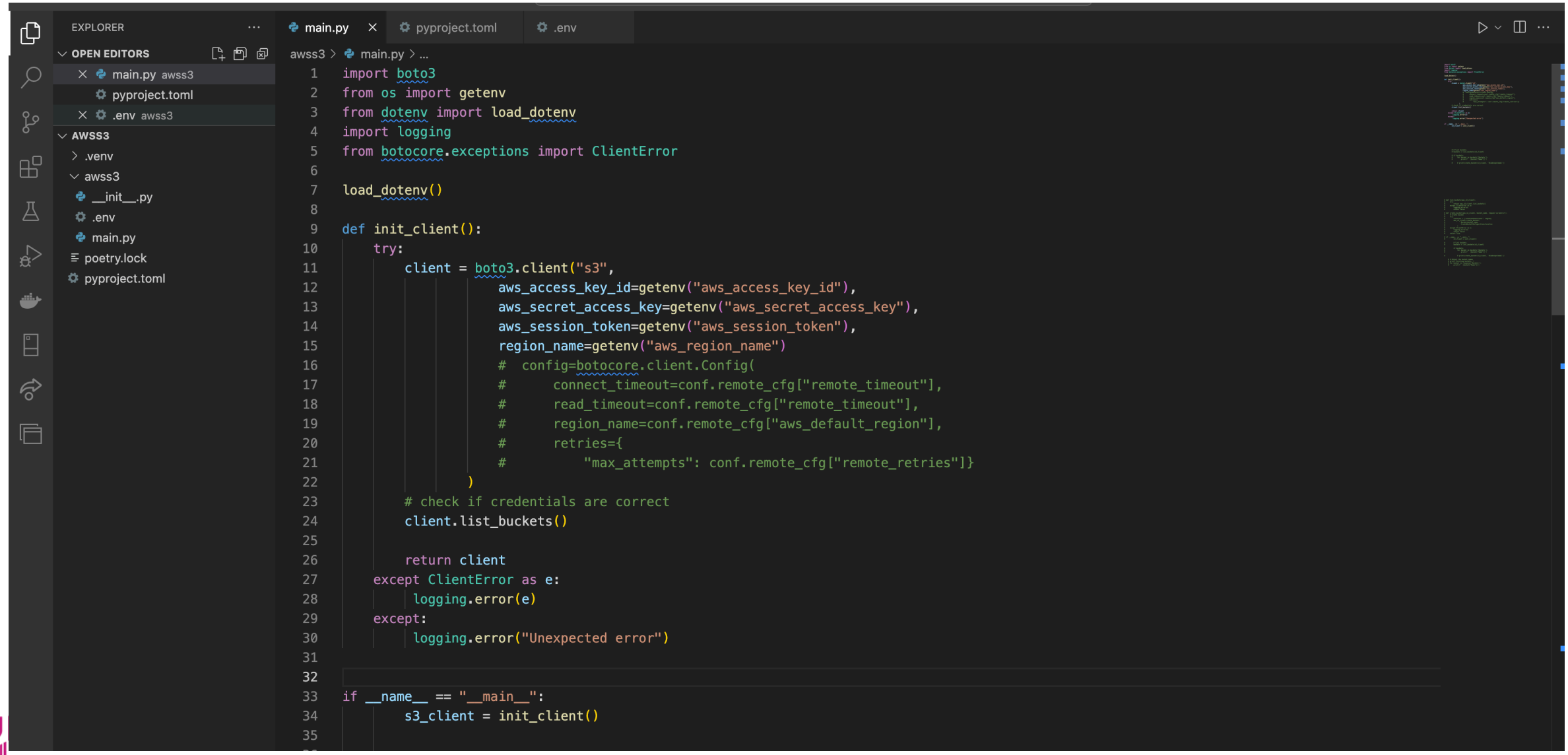
9. Access AWS Dashboard And create **S3** bucket.



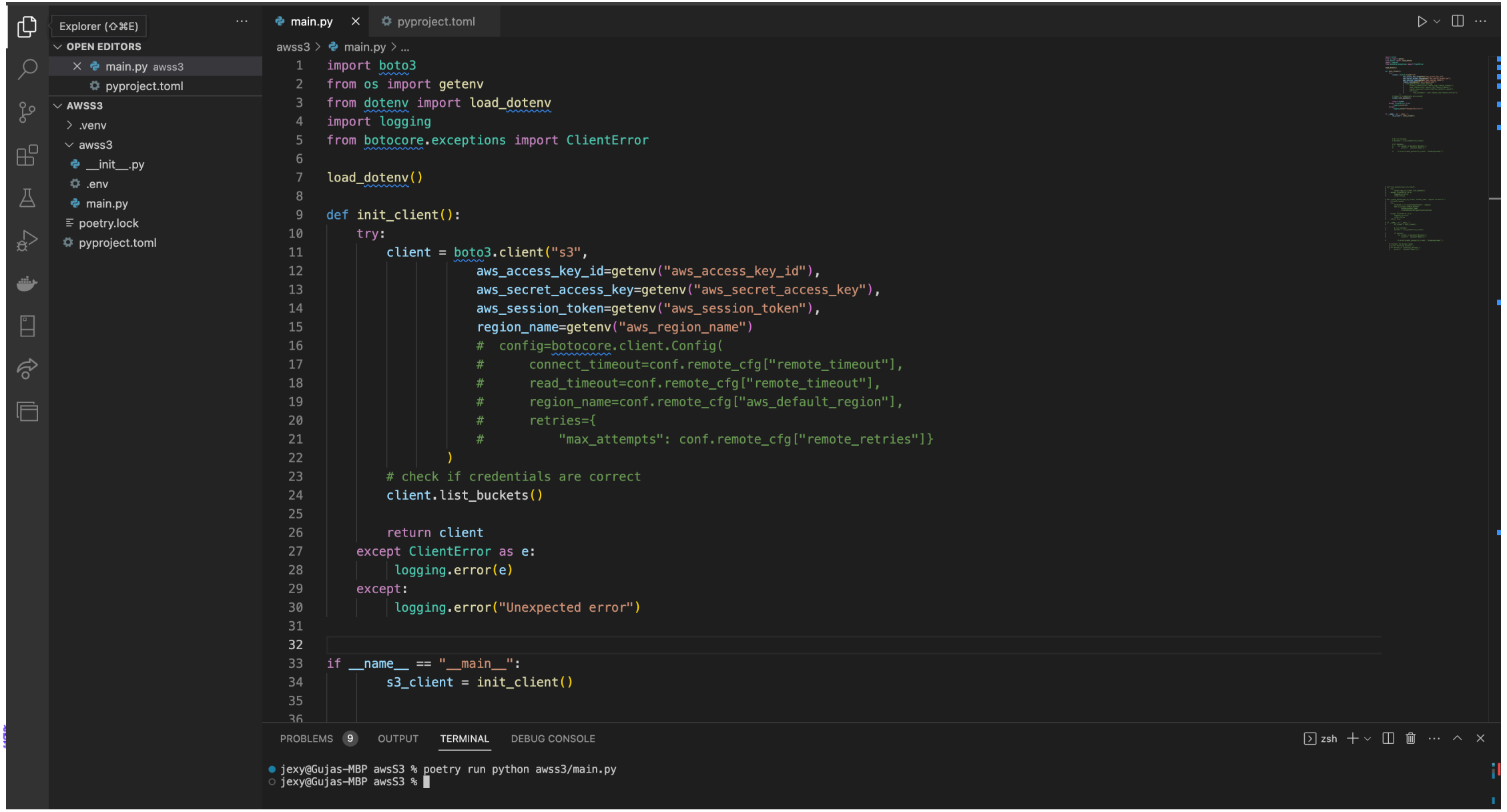
The same action using **AWS CLI:**

```
jexy@Gujas-MBP ~ % aws s3 mb s3://automatinawsbttu-commandline --profile=pythonBTUautomation
make_bucket: automatinawsbttu-commandline
jexy@Gujas-MBP ~ % aws s3 ls --profile pythonBTUautomation
2023-03-17 21:56:04 automatinawsbttu-commandline
2023-03-17 04:41:01 btudevopsteam1
2023-03-17 04:05:30 btugroup1
```

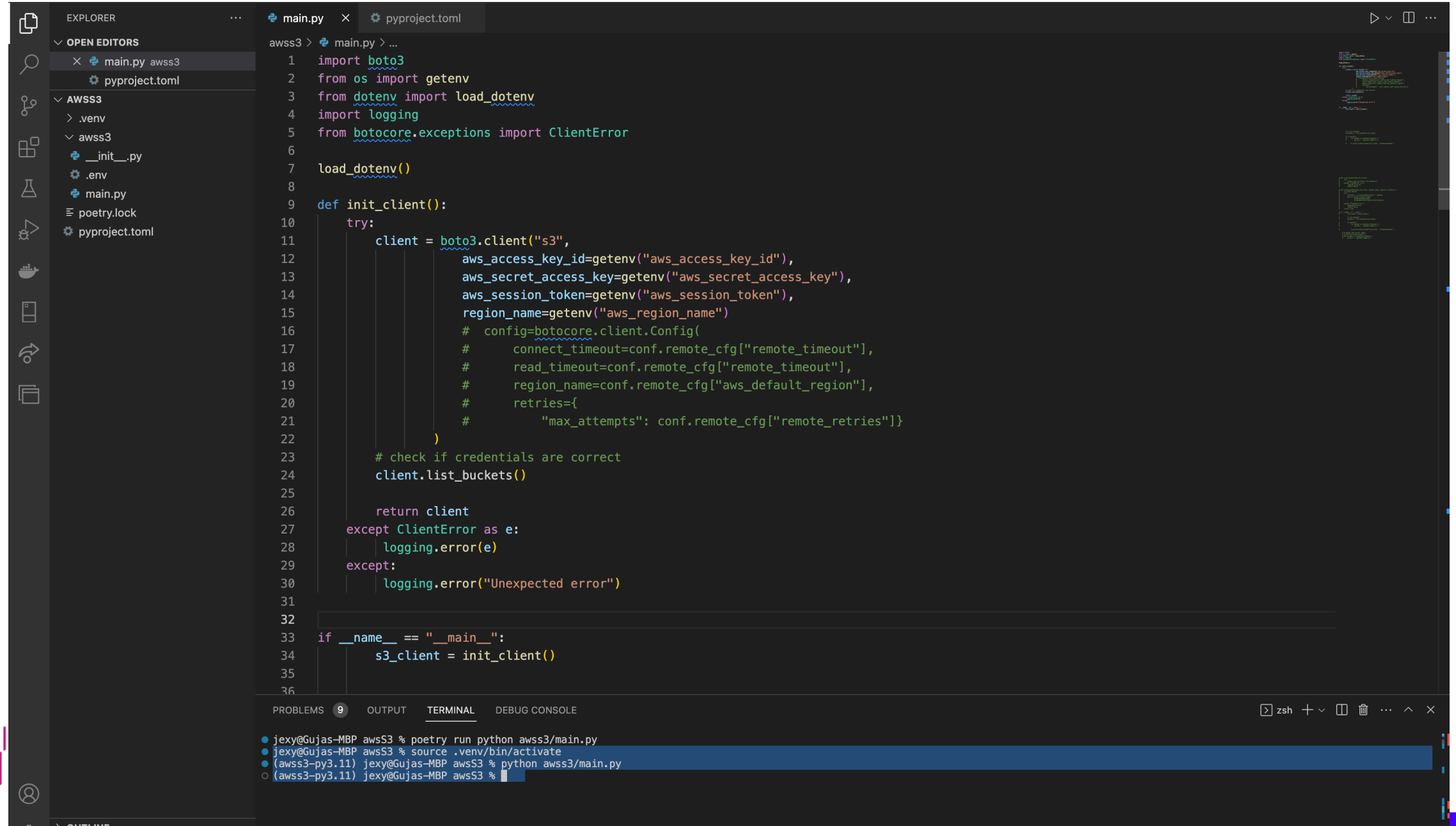
AWS SDK for Python (Boto3) S3
10. init_client()



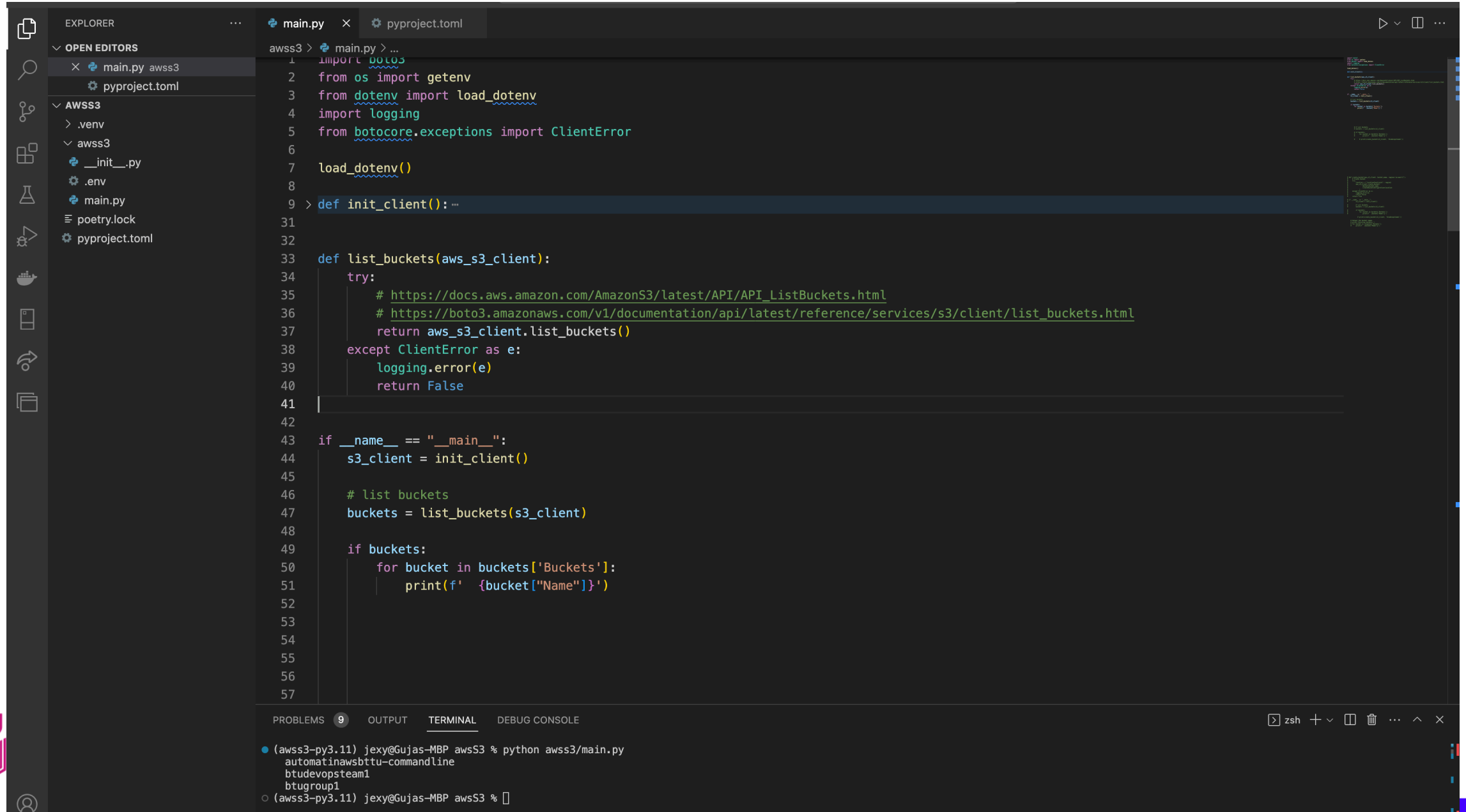
AWS SDK for Python (Boto3) S3
Run using poetry



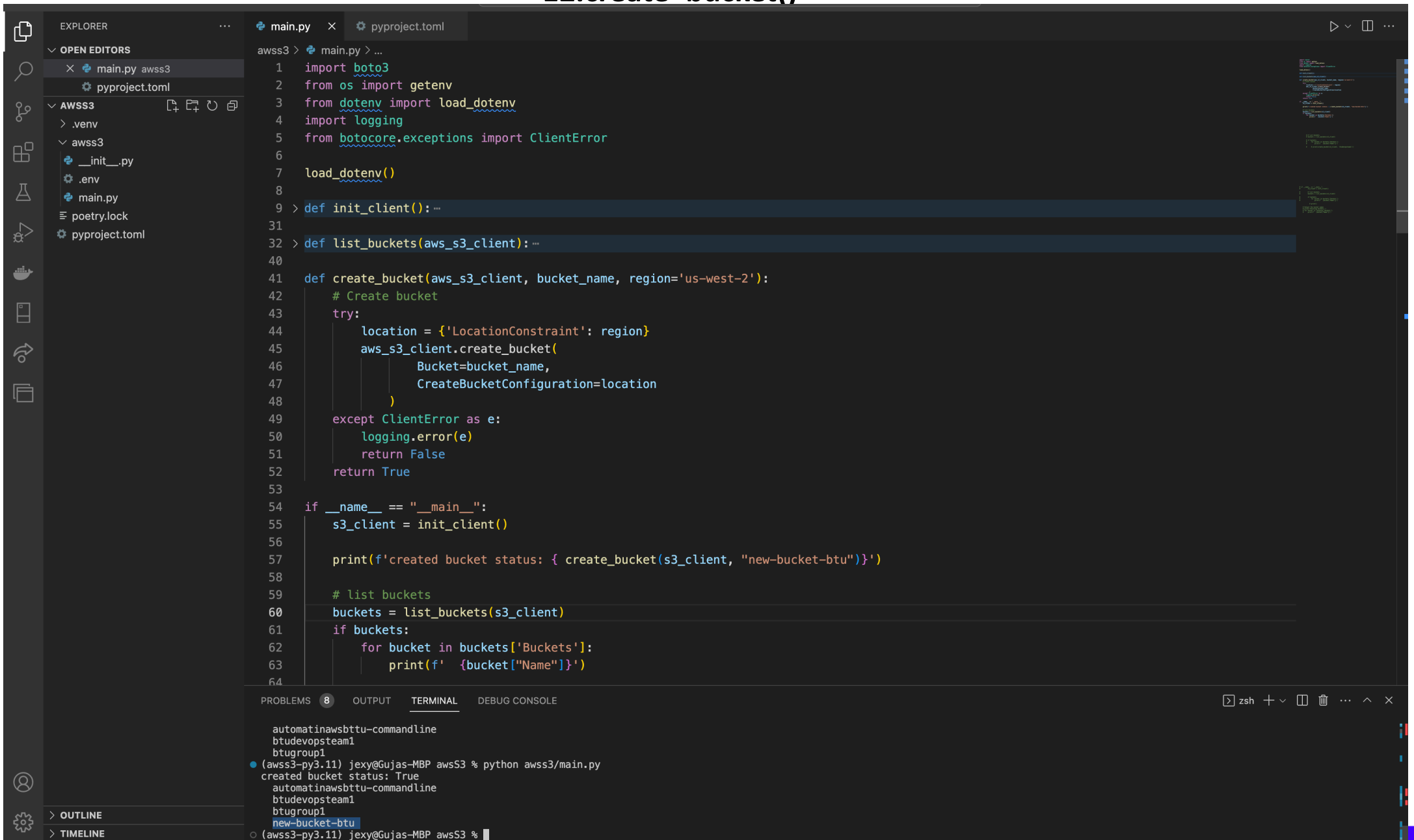
AWS SDK for Python (Boto3) S3
Run using poetry (activated virtualenv)



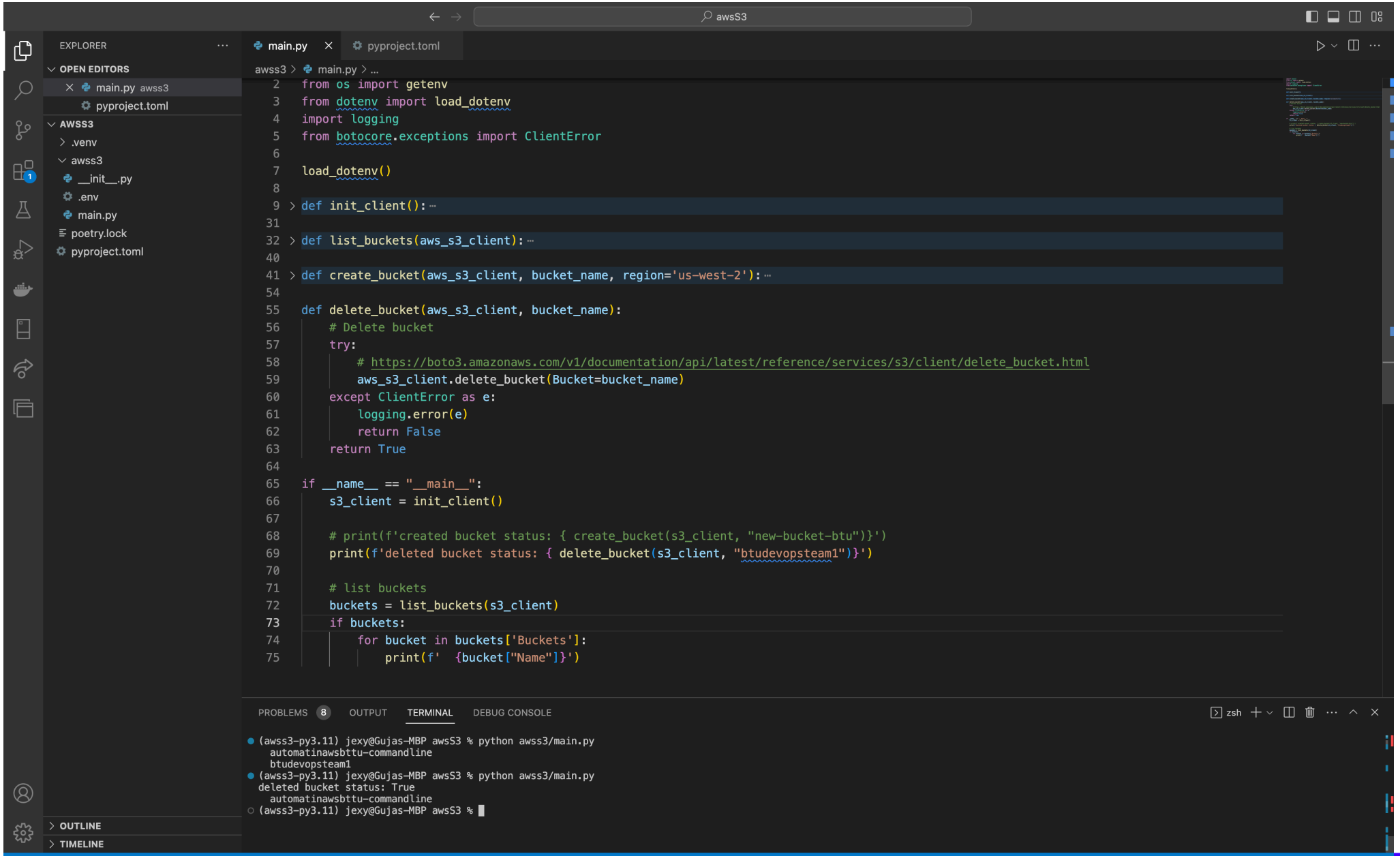
AWS SDK for Python (Boto3) S3
11.list_buckets()



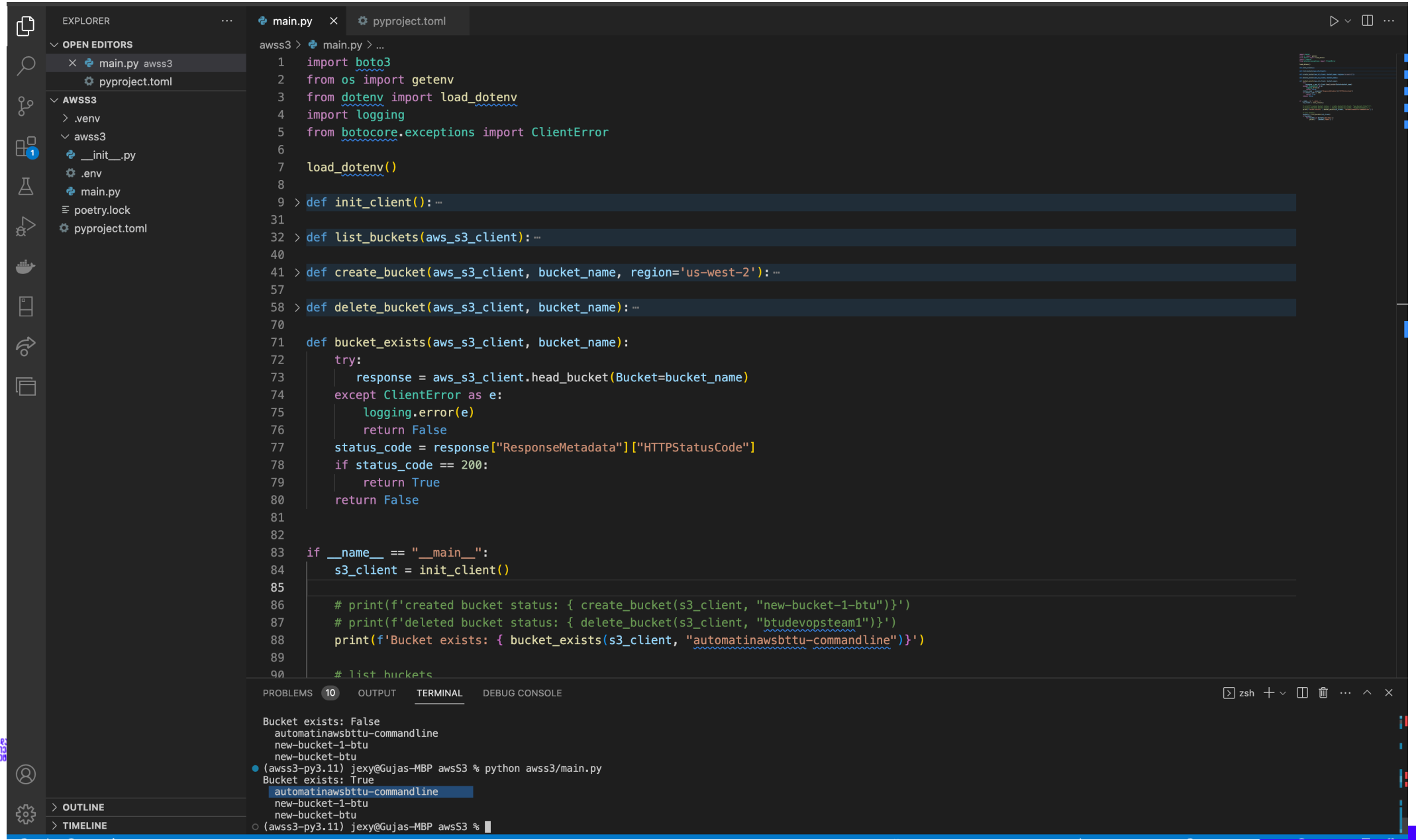
AWS SDK for Python (Boto3) S3
12.create_bucket()



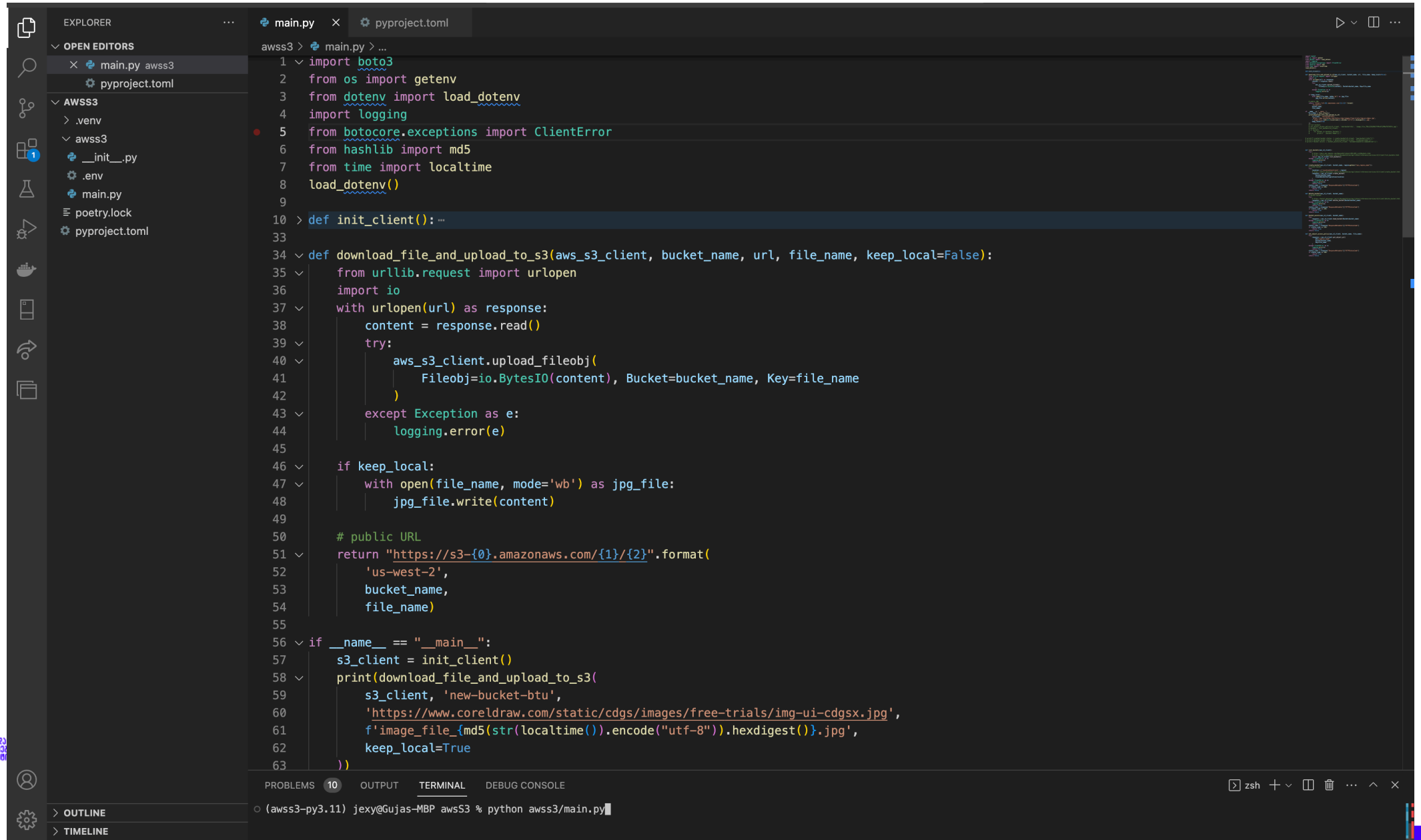
AWS SDK for Python (Boto3) S3
13. delete_bucket()



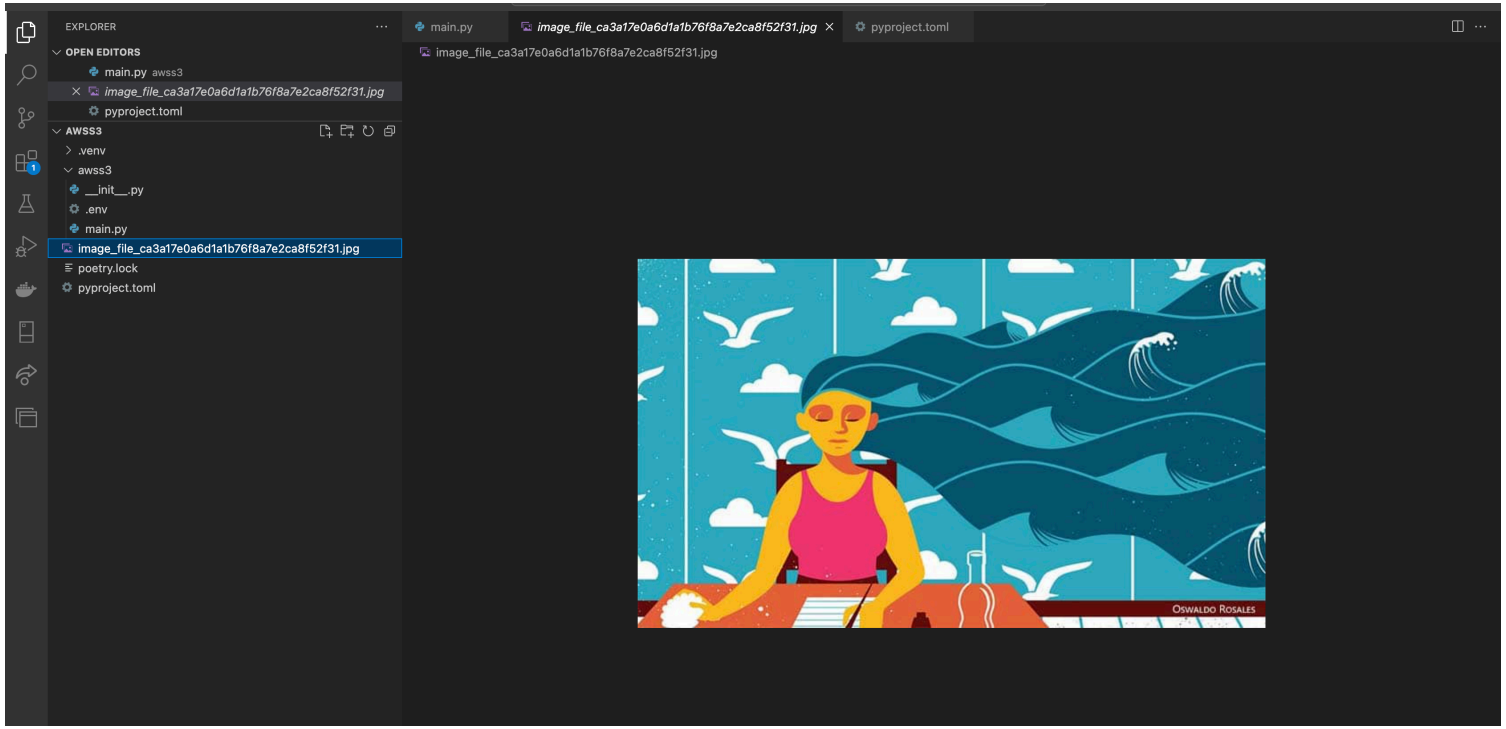
AWS SDK for Python (Boto3) S3
14. bucket_exists()



AWS SDK for Python (Boto3) S3
15. download_file_and_upload_to_s3()



AWS SDK for Python (Boto3) S3
No Read Permission :(



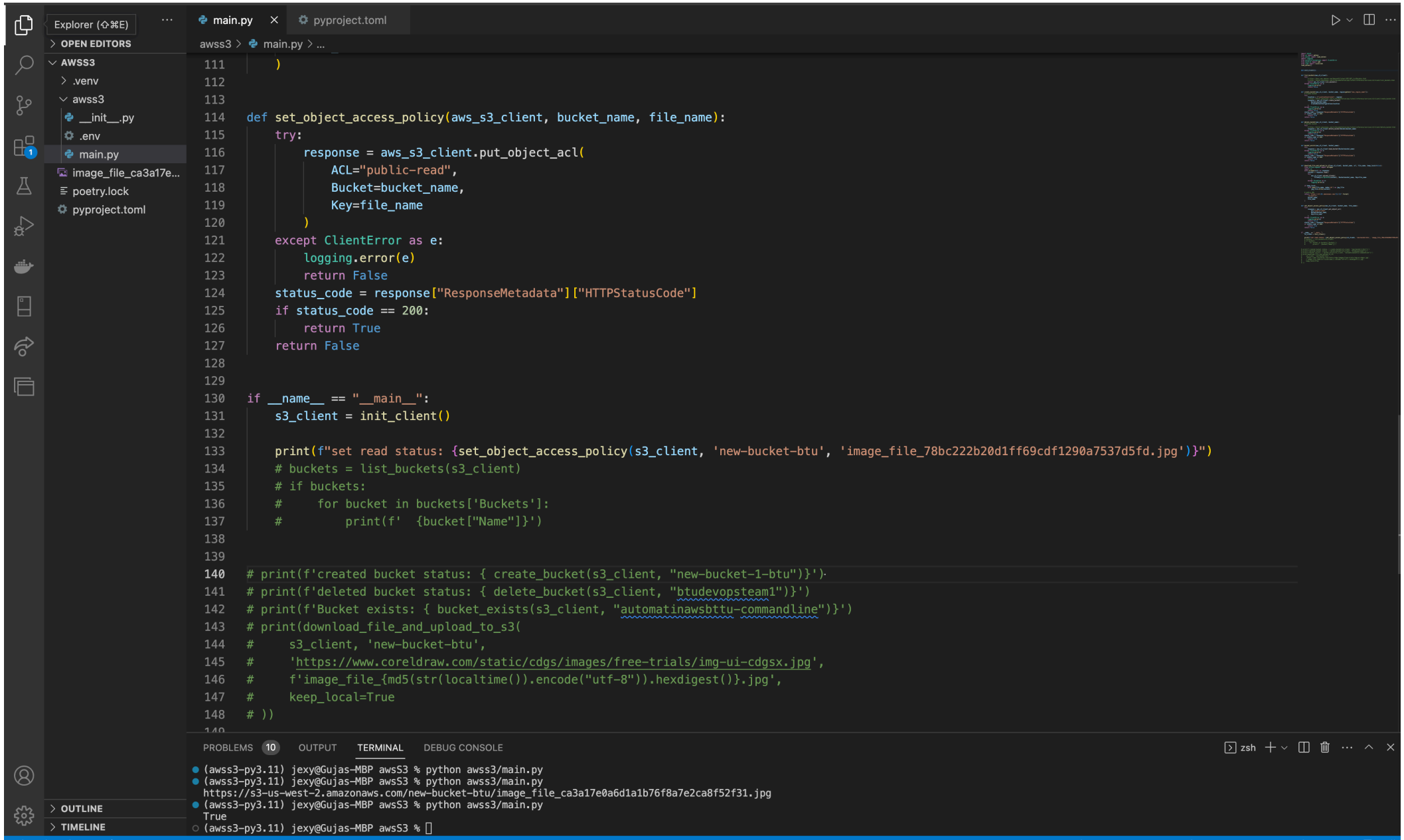

```
(awss3-py3.11) jexy@Gujas-MBP awsS3 % python awss3/main.py
https://s3-us-west-2.amazonaws.com/new-bucket-btu/image_file_ca3a17e0a6d1a1b76f8a7e2ca8f52f31.jpg
(awss3-py3.11) jexy@Gujas-MBP awsS3 %  $
```

s3-us-west-2.amazonaws.com

This XML file does not appear to have any style information associated with it. The document tree is shown below.


```
<Error>
  <Code>AccessDenied</Code>
  <Message>Access Denied</Message>
  <RequestId>SC8T2VB459A8XAXM</RequestId>
  <HostId>
    k2aC0t/5NaV73CMPNL79YS5pEreNlfMtakMT8uQigfB8h7VmS4bKvuKcNCG3AfE1AAtuh842vOs=
  </HostId>
</Error>
```

AWS SDK for Python (Boto3) S3
Grant Object Read Permission :)



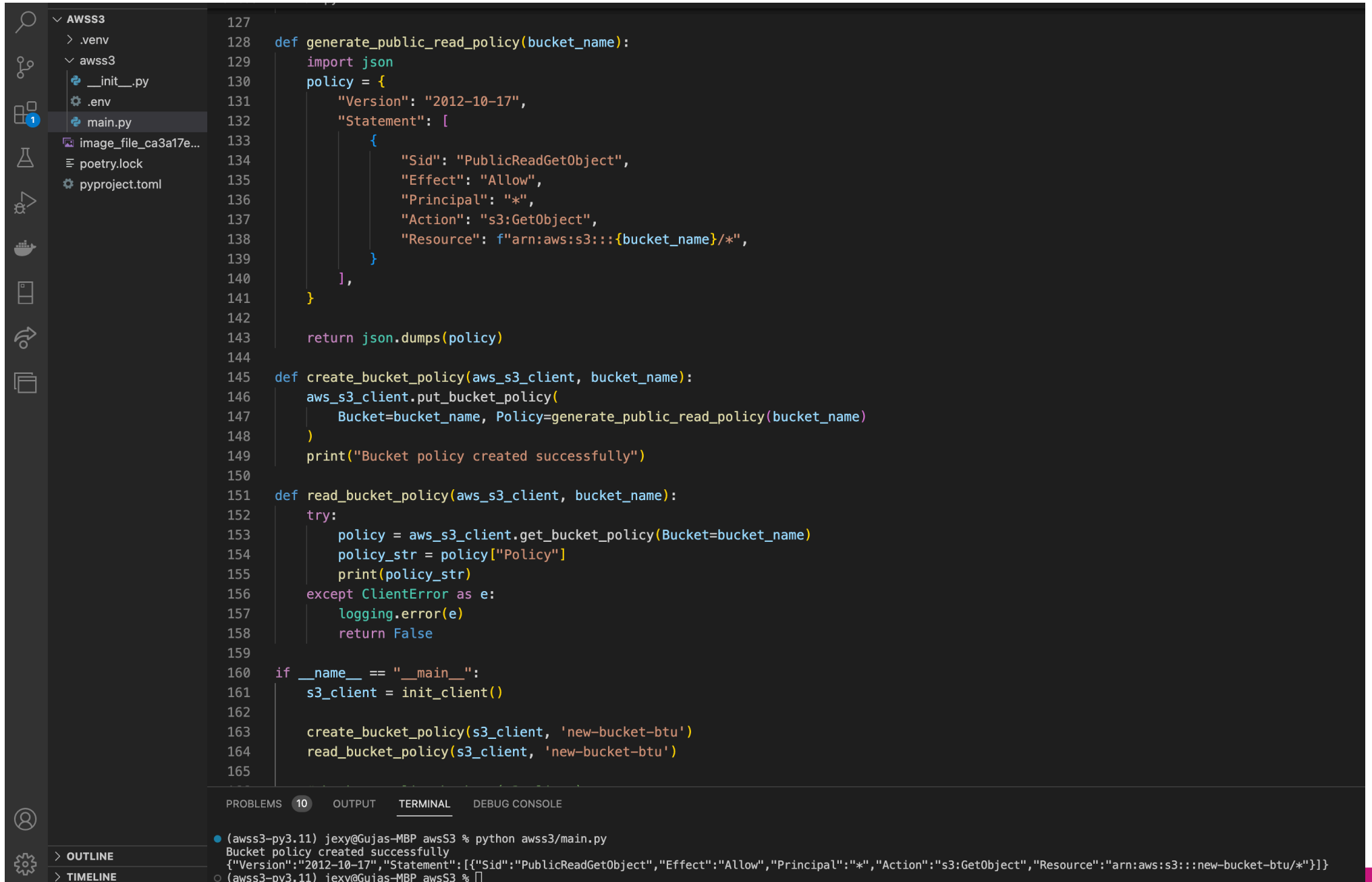
საცავის Policy-ის შექმნა

S3 საცავის Policy გვაძლევს შესაძლებლობას ვაკონტროლოთ წვდომები საცავზე და მასში არსებულ მონაცემებზე. მაგალითად, მივცეთ წაკითხვის უფლება ყველას, მივცეთ ჩაწერის უფლება კონკრეტული IP მისამართიდან და კიდევ ბევრი სხვა. მაგალითები შეგიძლიათ იხილოთ ბმულზე.

ჩვენ გავნიხილოთ საცავის გასაჯაროვების Policy

```
{"Version": "2012-10-17",
 "Statement": [
   {"Sid": "PublicReadGetObject",
     "Effect": "Allow",
     "Principal": "*",
     "Action": "s3:GetObject",
     "Resource": “arn:aws:s3:::BUCKET_NAME/*”}
 ]
}
```

საცავის Policy-ის შექმნა



Bonus Tasks

1. დაწერეთ პროგრამა, რომელსაც არგუმენტად გადაეცემა Bucket ის სახელი. პროგრამამ უნდა შეამოწმოს Bucket არსებობს თუ არა. თუ არსებობს უნდა დაწეროს რომ Bucket უკვე არსებობს, თუ არ არსებობს უნდა შექმნას ის.

2. დაწერეთ პროგრამა, რომელსაც არგუმენტად გადაეცემა Bucket ის სახელი. პროგრამამ უნდა შეამოწმოს Bucket ს გააჩნია თუ არა policy. თუ policy უკვე არსებობს უნდა დაბეჭდოს რომ policy უკვე არსებოს, წინააღმდეგ შემთხვევაში, უნდა შექმნას policy, რომელიც საჯაროდ წვდომადს გახდის ყველა ფაილს /dev და /test პრეფიქსების/ფაილების ქვეშ.

3. დაწერეთ პროგრამა, რომელსაც არგუმენტად გადაეცემა ბაკეტის სახელი. პროგრამამ უნდა შეამოწმოს ბაკეტი არსებობს თუ არა. თუ არსებობს უნდა წაშალოს, თუ არ არსებობს დაბეჭდოს რომ ბაკეტი არ არსებობს.

დაწერეთ კომფორტული **CLI** tool **S3** ტიპის საცავებთან სამუშაოდ.

გამოიყენეთ:

Poetry

dotenv

logging

boto3

მისაღებია ნებისმიერი CLI parsing library - ის განოყენება: Argparse, Docopt, Click, Typer

ჰქონდეს ყველა ფუნცია რაც აიხსნა ლექციაზე:

init_client()

list_buckets()

create_bucket()

delete_bucket()

bucket_exists()

download_file_and_upload_to_s3()

set_object_access_policy()

generate_public_read_policy()

create_bucket_policy()

read_bucket_policy()

***საბოლოო შედეგი უნდა იყოს S3 manager CLI tool ის განთავსება github - ზე შესაბამისი .readme ფაილით სადაც ახსნით ფუნქციონალს და აჩვენებთ თქვენს ტექნიკურ skill ებს - ჩართულობას პოტენციურ დამქირავებელს.**

Discussion

Hidden gem

Something possessing a value or beauty that is not immediately apparent, which therefore has received far less recognition than it deserves.

https://education.github.com/pack <- Hidden GEM 🏆

Recommended Tools For DevOps Engineers

Category	Tool
Operating System	Linux
Version Control	Git
Programming/Scripting	1. Python (Keep learning it parallelly with other tools) 2. Golang (Most DevOps Tools today are developed in Golang) 3. Linux Shell Scripting
Configuration Management	Ansible
Cloud Platform	AWS
Infratructure Provisioning	Terraform
Containerization Tool	Docker/Podman
Container Orchestration	Kubernetes
Continuous Integration	Jenkins/Github Actions
GitOps (Continuous Delivery)	ArgoCD/Flux CD
Logging & Monitoring	Prometheus/Grafana/Loki

ლიტერატურა

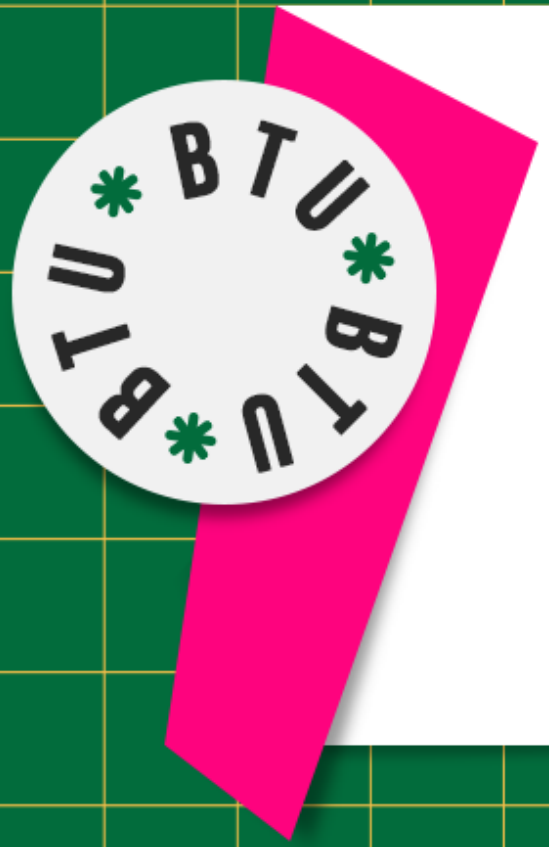
1. AWS Scripted: How to Automate the Deployment of Secure and Resilient Websites with Amazon Web Services VPC, ELB, EC2, RDS, IAM, SES and SNS - კრისტიან ჩერი, 2014წ. , 270 გვერდი
2. Amazon Web Services in Action - ანდრეას ვიტიგი, მიქაელ ვიტიგი, 2016წ., 424 გვ,
3. https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/s3.html

მადლობა ყურადღებისთვის!

კონტაქტი

თბილისი 0162, საქართველო
ი.ჭავჭავაძის გამზირი 82
INFO@BTU.EDU.GE
032 2 195 015
WWW.BTU.EDU.GE

გამოგვყევით:



ღრუბლოვანი სისტემების (AWS) ავტომატიზაცია Python-ის საშუალებით

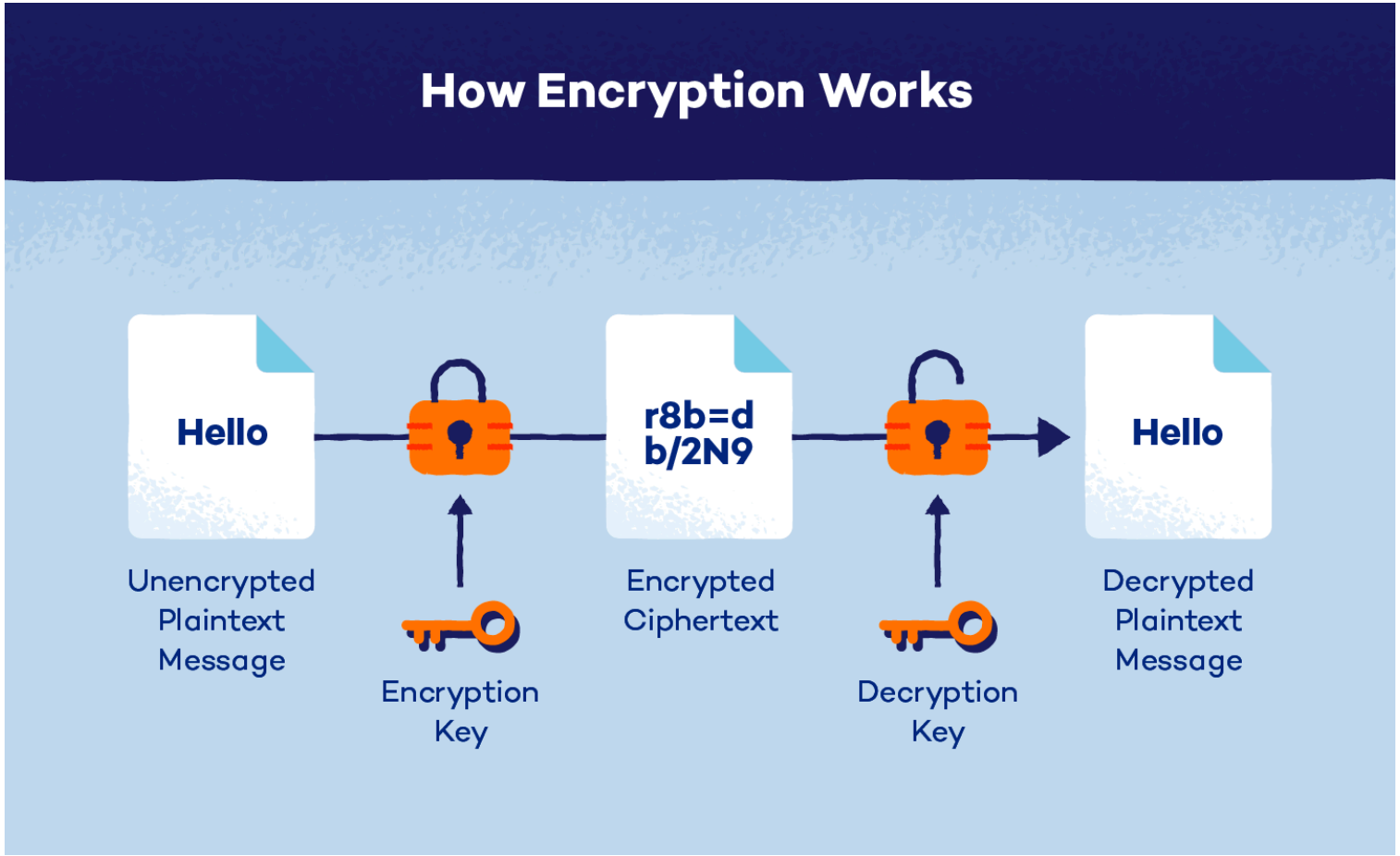
გუჯა ნემსაძე

საკითხები

Catch UP:

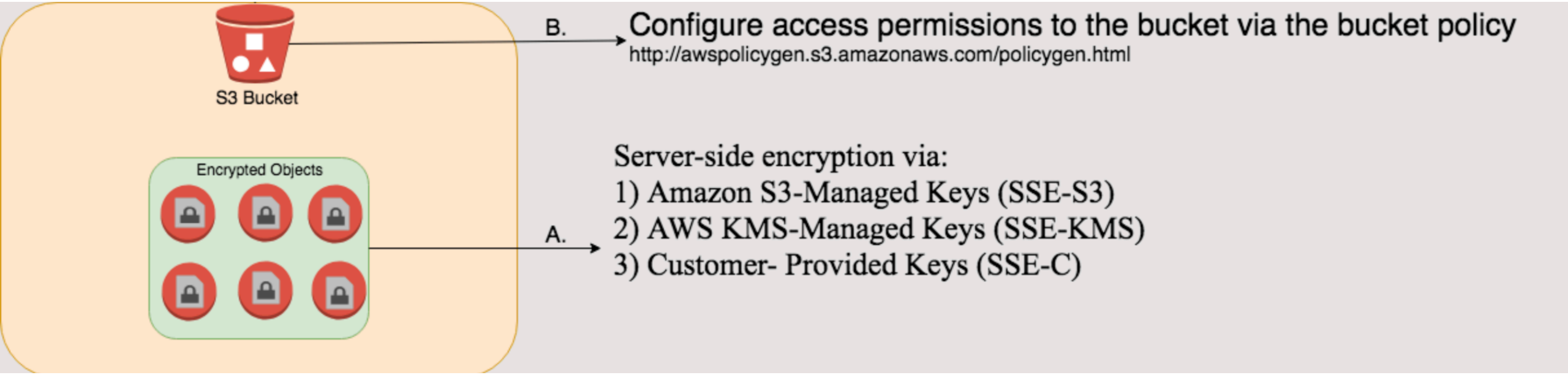
- Bucket-ის Server-Side დაშიფრვა
- საცავში(S3 Bucket) მცირე ზომის ფაილების ატვირთვა
- საცავში(S3 Bucket) დიდი ზომის ფაილების ატვირთვა (Multi-Part Upload),
- საცავში(S3 Bucket) არსებული ობიექტების და ფაილე-ბის წაკითხვა
- საცავში(S3 Bucket) ვერსიების მართვის ორგანიზება
- საცავის(S3 Bucket) “Lifecycle Policy“-ს კონფიგურირება

Bucket-ის Server-Side დაშიფრვა



https://www.microfocus.com/en-us/what-is/encryption

Bucket-ის Server-Side დაშიფრვა



https://www.learnaws.org/2022/10/09/aws-s3-server-side-encryption/#aws-s3-encryption

- **Amazon S3 buckets have bucket encryption enabled by default, and new objects are automatically encrypted by using server-side encryption with Amazon S3 managed keys (SSE-S3). This encryption applies to all new objects in your Amazon S3 buckets, and comes at no cost to you.**
- **Enabling server-side encryption on S3 buckets at the object level protects data at rest and helps prevent the breach of sensitive information assets.**
- **Companies who are looking for better protection for data at rest and those who need to comply with regulations like HIPAA or data breach notification laws**

Bucket-ის Server-Side დაშიფვრა

```python
def set_bucket_encryption(aws_s3_client, bucket_name):
    response = aws_s3_client.put_bucket_encryption(
        Bucket=bucket_name,
        ServerSideEncryptionConfiguration={
            "Rules": [
                {"ApplyServerSideEncryptionByDefault": {"SSEAlgorithm": "AES256"}}
            ]
        },
    )
    status_code = response["ResponseMetadata"]["HTTPStatusCode"]
    if status_code == 200:
        return True
    return False
```

Bucket-ის Server-Side დაშიფრვა

```python
def read_bucket_encryption(aws_s3_client, bucket_name):
    return aws_s3_client.get_bucket_encryption(Bucket=bucket_name)

```

მცირე ზომის ფაილების ატვირთვა

```python
def upload_file(aws_s3_client, filename, bucket_name):
    response = aws_s3_client.upload_file(filename, bucket_name, "hello.txt")

    status_code = response["ResponseMetadata"]["HTTPStatusCode"]
    if status_code == 200:
        return True
    return False


def upload_file_obj(aws_s3_client, filename, bucket_name):
    with open(filename, "rb") as file:
        aws_s3_client.upload_fileobj(file, bucket_name, "hello_obj.txt")


def upload_file_put(aws_s3_client, filename, bucket_name):
    with open(filename, "rb") as file:
        aws_s3_client.put_object(Bucket=bucket_name, Key="hello_put.txt", Body=file.read())
```

დიდი ზომის ფაილების ატვირთვა

```python
import os
from os import getenv

import boto3
from botocore.config import Config

AWS_REGION = getenv("AWS_REGION", "eu-central-1")
CUSTOM_CONFIG = Config(
    region_name=AWS_REGION,
)
BUCKET = "btu-lecture-3"
s3_client = boto3.client("s3", config=CUSTOM_CONFIG)
PART_BYTES = 1024 * 10


def multipart_upload(filename, key):
    mpu = s3_client.create_multipart_upload(Bucket=BUCKET, Key=key)
    mpu_id = mpu["UploadId"]

    parts = []
    uploaded_bytes = 0
    total_bytes = os.stat(filename).st_size
```

დიდი ზომის ფაილების ატვირთვა

```
    with open(filename, "rb") as f:
        i = 1
        while True:
            data = f.read(PART_BYTES)
            if not len(data):
                break
            part = s3_client.upload_part(Body=data, Bucket=BUCKET, Key=key, UploadId=mpu_id,
PartNumber=i)
            parts.append({"PartNumber": i, "ETag": part["ETag"]})
            uploaded_bytes += len(data)
            print("{0} of {1} uploaded".format(uploaded_bytes, total_bytes))
            i += 1

    result = s3_client.complete_multipart_upload(
        Bucket=BUCKET, Key=key, UploadId=mpu_id, MultipartUpload={"Parts": parts}
    )
    print(result)

    return result


if __name__ == "__main__":
    multipart_upload("test.txt", "test_multipart.txt")
```

არსებული ობიექტების წაკითხვა

```
def read_objects():
    result = s3_client.list_objects(Bucket=BUCKET)
    for obj in result.get("Contents", []):
        print(obj.get("Key"), obj.get("Size"))


def download_file(key):
    s3_client.download_file(BUCKET, key, "result.txt")


def main():
    download_file("hello_obj.txt")


if __name__ == "__main__":
    main()
```

ვერსიები საცავში

ობიექტს საცავში შეიძლება ჰქონდეს ვერსიები. ამისათვის საცავს უნდა ჩავურთოთ ვერსიები ფუნქცია. ამის შემდეგ უკვე შეგვიძლია ერთი და იგივე ობიექტი გვქონდეს რამდენიმე ვერსიად

```
def enable_versioning():
    response = s3_client.put_bucket_versioning(
        Bucket=BUCKET,
        VersioningConfiguration={
            "Status": "Enabled",
        },
    )

    print(response)
```

“Lifecycle Policy“-ს კონფიგურირება

იმისათვის, რომ ეფექტურად ვმართოთ ობიექტები საცავში, შეგვიძლია გამოვიყენოთ საცავის ციკლის კონფიგურაცია. საცავის ციკლის კონფიგურაცია არის წესების ერთობლიობა, რომელიც განსაზღვრავს თუ რა მოქმედებები უნდა ჩატარდეს ობიექტების ჯგუფებზე. გვაქვს ორი ტიპის ქმედება:

- **გადატანის ქმედება(Transition Actions): ეს ქმედებები გასაზღვრავს როდის უნდა გადავიდეს ობიექტი სხვა კლასის საცავში. მაგალითად, შეგვიძლია დავწეროთ კონფიგურაცია რომელიც ობიექტს შექმნიდან 30 დღეში გადაიტანს Standart-IA კალსიდან S3 Glacier კლასში.**
- **ვადის გასვლის ქმედება(Expiration Actions): ამ ტიპის ქმედებები შეგვიძლია გამოვიყენოთ ობიექტების წასაშლელად. მაგალითად, შევქმნათ კონფიგურაცია რომელიც ობიექტის შექმნიდან 120 დღის გასვლის შემდეგ წაშლის ობიექტს.**

boto3 მოდულის გამოყენებით შეგვიძლია დავაკონფიგურიროთ საცავის ციკლი. მეთოდის სრული დოკუმენტაცია შეგიძლიათ იხილოთ აქ

“Lifecycle Policy“-ს კონფიგურირება

```
def put_policy():
    lfc = {
        "Rules": [
            {
                "Expiration": {"Days": 7},
                "ID": "devobjects",
                "Filter": {"Prefix": "dev"},
                "Status": "Enabled",
            }
        ]
    }
    s3_client.put_bucket_lifecycle_configuration(
        Bucket="bucket", LifecycleConfiguration=lfc
    )


def main():
    put_policy()
```

ლიტერატურა

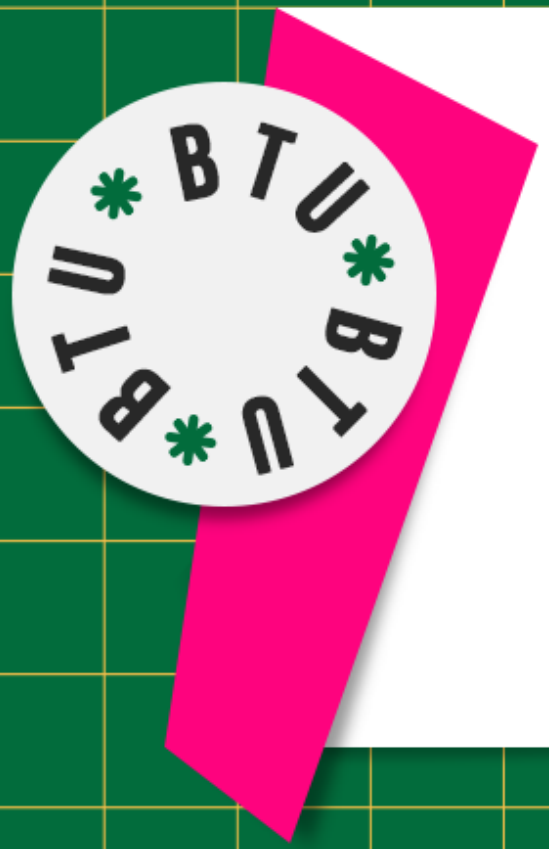
1. AWS Scripted: How to Automate the Deployment of Secure and Resilient Websites with Amazon Web Services VPC, ELB, EC2, RDS, IAM, SES and SNS - კრისტიან ჩერი, 2014წ. , 270 გვერდი
2. Amazon Web Services in Action - ანდრეას ვიტიგი, მიქაელ ვიტიგი, 2016წ., 424 გვ,
3. https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/s3.html

მადლობა ყურადღებისთვის!

კონტაქტი

თბილისი 0162, საქართველო
ი.ჭავჭავაძის გამზირი 82
INFO@BTU.EDU.GE
032 2 195 015
WWW.BTU.EDU.GE

გამოგვყევი:



ღრუბლოვანი სისტემების (AWS) ავტომატიზაცია Python-ის საშუალებით

გუჯა ნემსაძე

საკითხები

Catching UP - 2

- შუალედური გამოცდების პერიოდი იწყება **23 აპრილიდან**.

- Multipart Upload
- Amazon S3 multipart upload limits
- S3 Versioning
    - When to use
- Tasks

შუალედური გამოცდების პერიოდი იწყება **23 აპრილიდან**.

- სულ იქნება 2 ვარიანტი.
- მაქსიმალური შეფასება **30** ქულა.
  - **10** ქულით შეფასდება თეორიული.
  - **20** ქულით პრაქტიკული ნაწილი.
- შუალედურზე შევა მხოლოდ განვლილი მასალა.
- გექნებათ საშუალება გამოიყენოთ ინტერნეტი.
- გამოცდას ჩაატარებს საგამოცდო ცენტრი.
- სულ შუალედურს დაეთმობა 60 წუთი, ამოცანების სირთულეც იქნება შესაბამისად შერჩეული რათა 60 წუთში ჩაერტიოთ.
- ყველა ნაშრომი რომელიც შეფასდება დუბლიკატად გაუქმდება.
- გამოცდამდე აუცილებლად დავუთმობთ ლექციის დროს გადამეორებას რათა გამოცდაზე თავდაჯერებულად გახვიდეთ.

Multipart Upload

- This technique allows you to split a file into several small chunks and upload them all sequentially or in parallel, empowering you to deal with large files concisely.
- **Improved throughput** – You can upload parts in parallel to improve throughput.
- **Quick recovery from any network issues** – Smaller part size minimizes the impact of restarting a failed upload due to a network error.
- **Pause and resume object uploads** – You can upload object parts over time. After you initiate a multipart upload, there is no expiry; you must explicitly complete or stop the multipart upload.
- **Begin an upload before you know the final object size** – You can upload an object as you are creating it.

https://docs.aws.amazon.com/AmazonS3/latest/userguide/mpuoverview.html

Amazon S3 multipart upload limits

Item	Specification
Maximum object size	5 TiB
Maximum number of parts per upload	10,000
Part numbers	1 to 10,000 (inclusive)
Part size	5 MiB to 5 GiB. There is no minimum size limit on the last part of your multipart upload.
Maximum number of parts returned for a list parts request	1000
Maximum number of multipart uploads returned in a list multipart uploads request	1000

https://docs.aws.amazon.com/AmazonS3/latest/userguide/qfacts.html

S3 Versioning

- S3 provides serious durability, it does not protect you from overwriting your objects or even deleting those objects. Or does it? Not by default, but it does if we enable **versioning**.
- **Audit trail.** You can demonstrate to authorities, accountants etc., how a file (strictly, object) has changed over time, proving it has not been falsified subsequently—or that it has
- If you use S3 for storing logs, it's just about **impossible for hackers to "fix up" old logs** to hide their interference without leaving a breadcrumb trail.
- Being able to retrieve a specific data set from a given point in **time**.

S3 Versioning - When to use

If your data is more important than the cost then it's really good to have a version-enabled bucket.

Don't forget:

- Once you enable versioning, you cannot completely disable it.
- As you are storing multiple versions of the same object, your bill might go up.

https://acloudguru.com/blog/engineering/amazon-s3-versioning-what-how-why

Tasks

1. Create new bucket and upload file with using **multipart_upload**, enable read permission **only to the uploaded file** and **not** the whole bucket.
2. Enable Versioning on newly created bucket.
3. Upload several versions of file.
4. Rollback file to first version.
5. Host static website https://boto3.amazonaws.com/v1/documentation/api/latest/guide/s3-example-static-web-host.html

ლიტერატურა

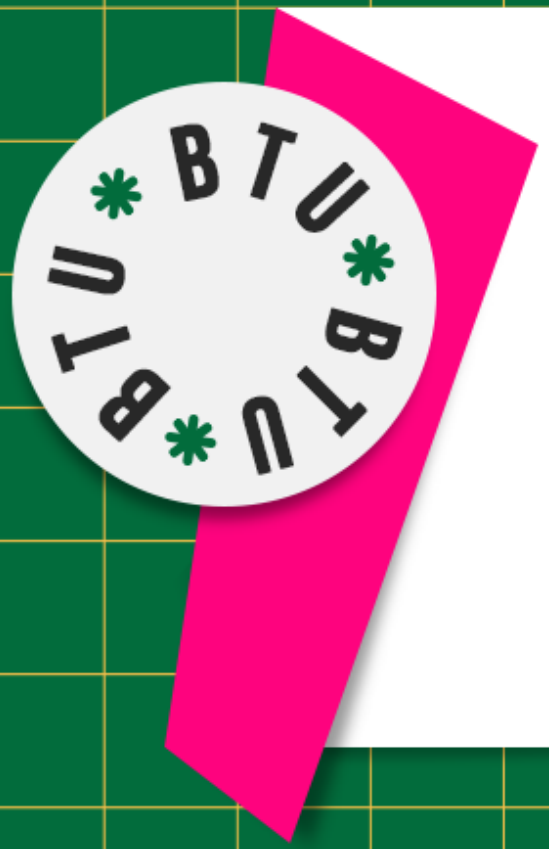
1. AWS Scripted: How to Automate the Deployment of Secure and Resilient Websites with Amazon Web Services VPC, ELB, EC2, RDS, IAM, SES and SNS - კრისტიან ჩერი, 2014წ. , 270 გვერდი
2. Amazon Web Services in Action - ანდრეას ვიტიგი, მიქაელ ვიტიგი, 2016წ., 424 გვ,
3. https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/s3.html

მადლობა ყურადღებისთვის!

კონტაქტი

თბილისი 0162, საქართველო
ი.ჭავჭავაძის გამზირი 82
INFO@BTU.EDU.GE
032 2 195 015
WWW.BTU.EDU.GE

გამოგვყევით:



ღრუბლოვანი სისტემების (AWS) ავტომატიზაცია Python-ის საშუალებით

Guja Nemsadze

საკითხები

- S3-CLI script with static web page hosting.
- AWS Lambda.
- s3 ტრიგერი.
- მონაცემთა დამუშავების არხი (Serverless data processing pipeline).

Steps To Host Static Web Page

- Create Bucket.
- Give bucket read permission policy - “s3:GetObject”.
- Set “WebsiteConfiguration”.
- Upload your “index.html” file with its dependencies.
- Access the web page.

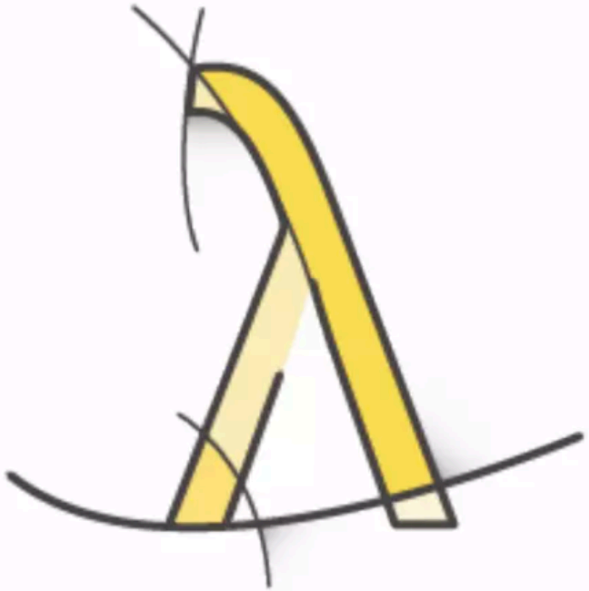
Task

Host Static Web Page With Dependency

Using custom CLI

AWS Lambda

Serverless, event-driven compute service



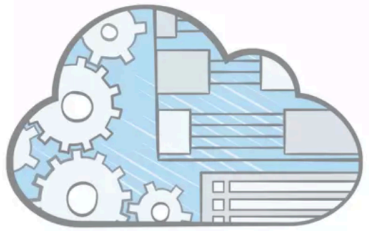
Lambda = microservice without servers

AWS Lambda

AWS Lambda – Benefits

1

SERVERLESS



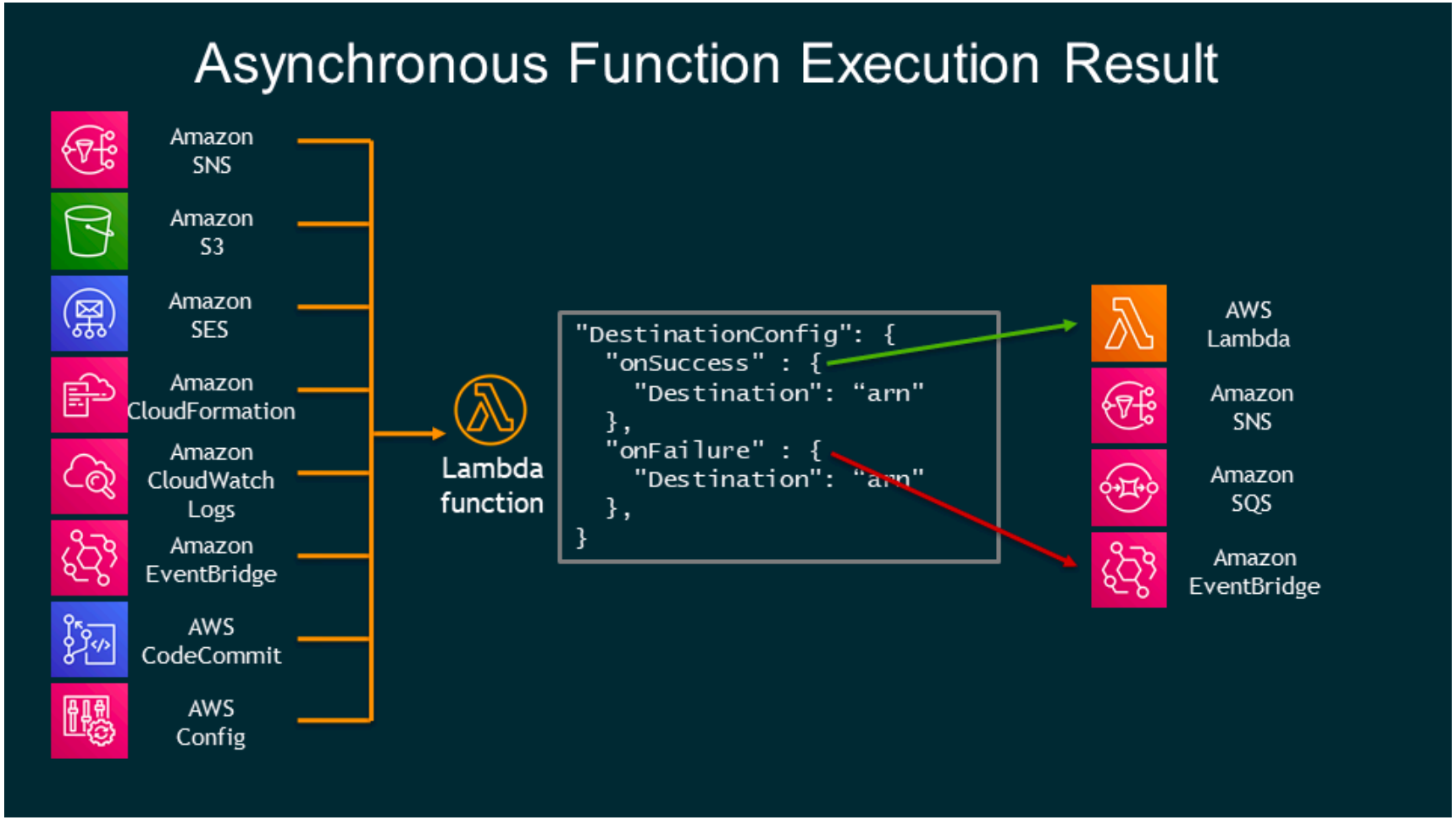
2

EVENT-DRIVEN SCALE

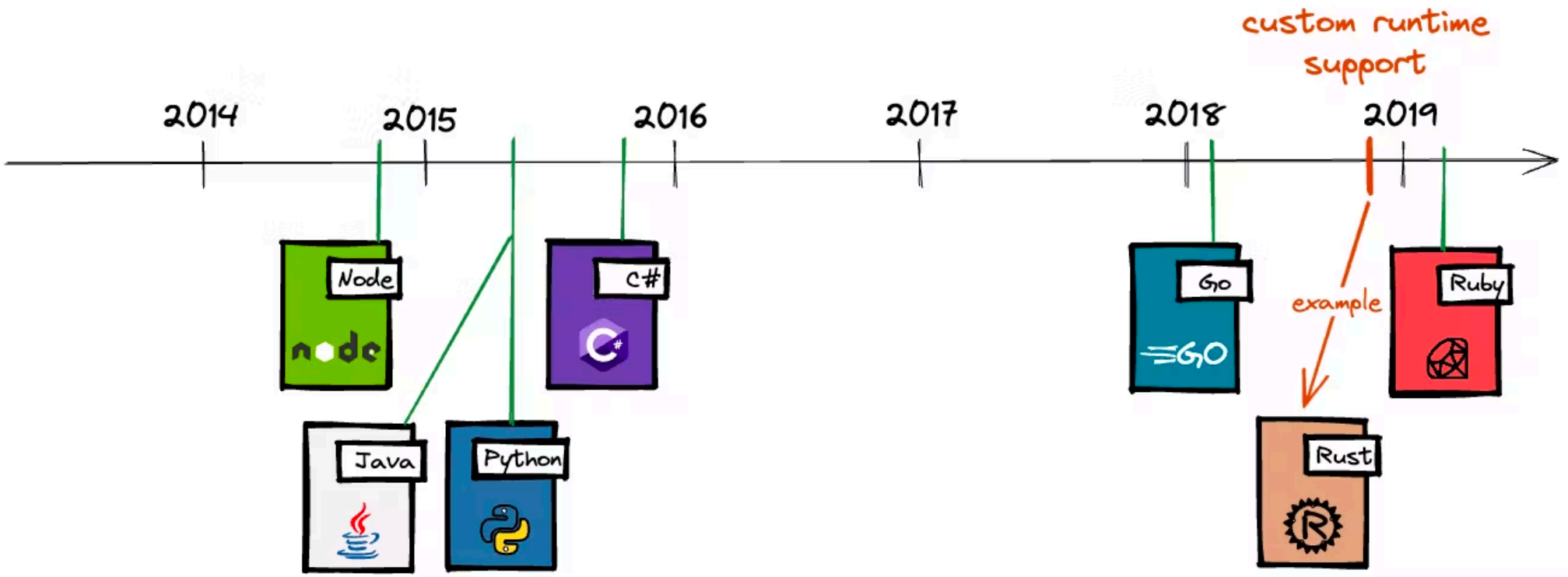
3

SUBSECOND BILLING

AWS Lambda Integrations



AWS Lambda supported runtimes:



AWS Lambda: use cases

- Operating serverless websites
- Rapid document conversion
- Predictive page rendering
- Working with external services
- Log analysis on the fly
- Automated backups and everyday tasks
- Processing uploaded S3 objects
- Backend cleaning
- Bulk real-time data processing

https://www.contino.io/insights/aws-lambda-use-cases#:~:text=Lambda%20performs%20all%20the%20operational,monitoring%20and%20logging%20your%20code.

AWS Lambda limits

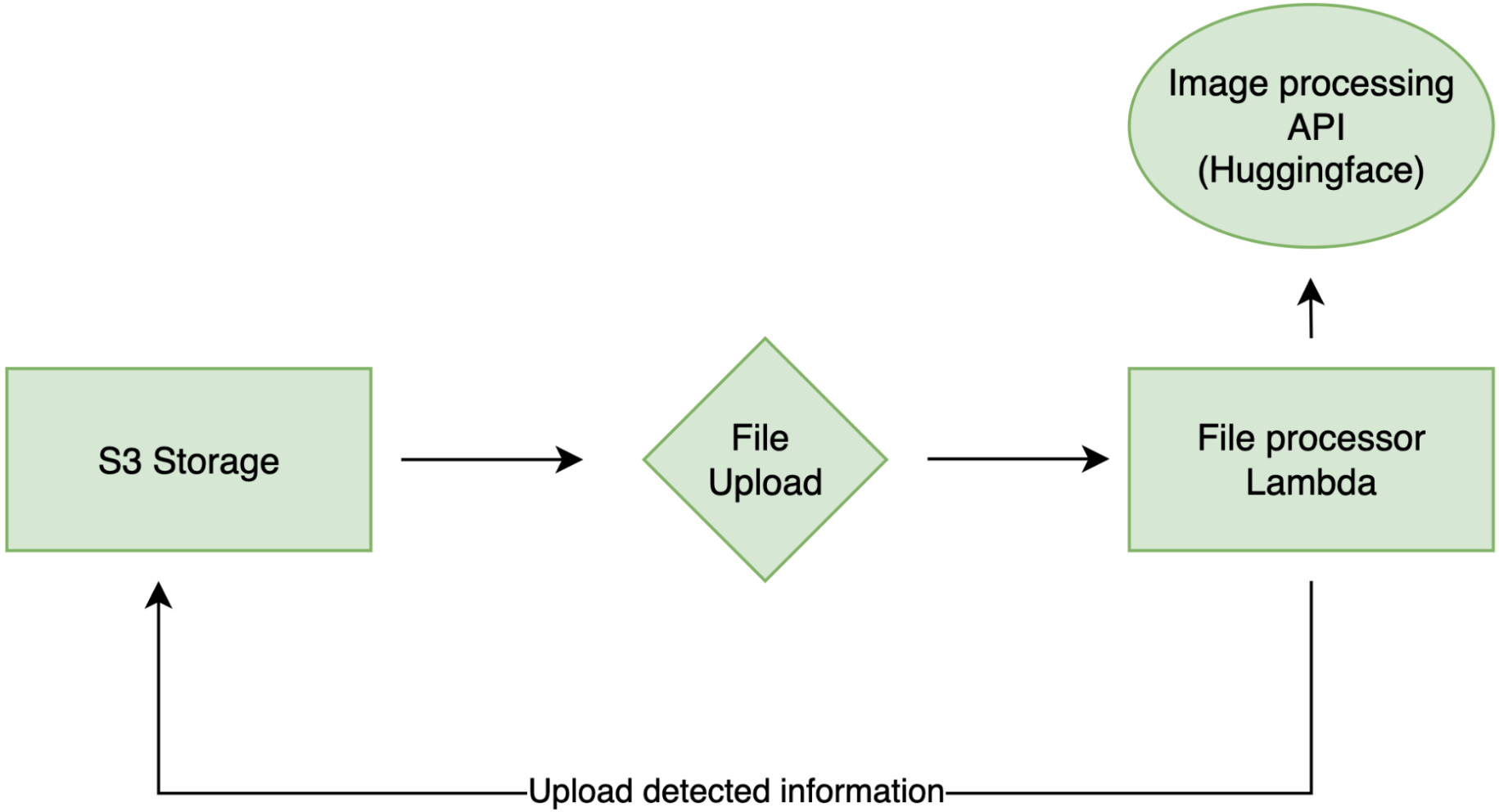
There are some “hard limitations” for the runtime environment: **the disk space** is limited to **500MB**, memory can vary from **128MB to 3GB** and the **execution timeout for a function is 15 minutes**. Package constraints like the size of **deployment package (250MB)** and the number of file descriptors (1024) are also defined as hard limits.

Similarly, there are “limitations” for the requests served by Lambda: **request and response body** synchronous event payload can be a **maximum of 6 MB** while an asynchronous invocation payload can be up to 256KB. At the moment, the only soft “limitation”, which you can request to be increased, is the number of concurrent executions, which is a safety feature to prevent any accidental recursive or infinite loops go wild in the code. This would throttle the number of parallel executions.

https://docs.aws.amazon.com/lambda/latest/dg/gettingstarted-limits.html

https://mikhail.io/serverless/coldstarts/aws/

მონაცემთა დამუშავების არხი (Serverless data processing pipeline)



[Lambda Code]

ლიტერატურა

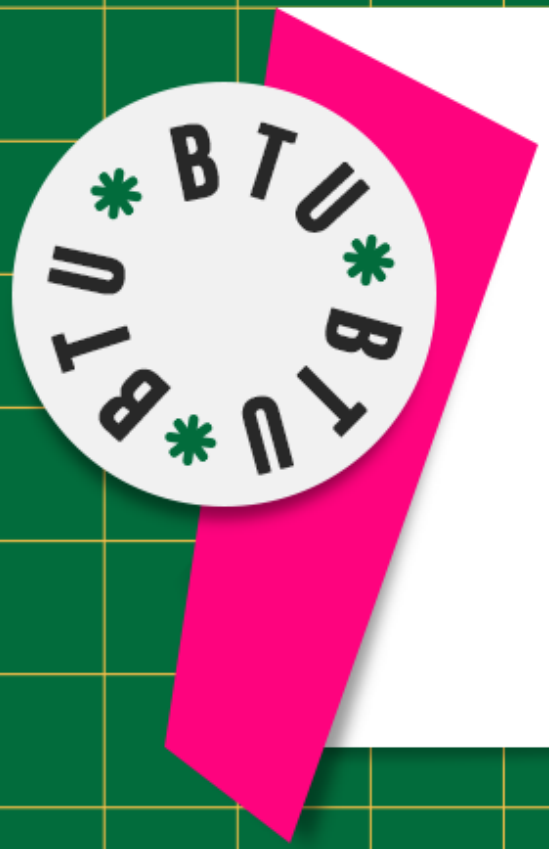
1. AWS Scripted: How to Automate the Deployment of Secure and Resilient Websites with Amazon Web Services VPC, ELB, EC2, RDS, IAM, SES and SNS - კრისტიან ჩერი, 2014წ. , 270 გვერდი
2. Amazon Web Services in Action - ანდრეას ვიტიგი, მიქაელ ვიტიგი, 2016წ., 424 გვ,
3. https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/lambda.html
4. https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/iam.html
5. https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/s3.html#S3.Client.put_bucket_notification_configuration
6. https://api-inference.huggingface.co/docs/python/html/detailed_parameters.html#object-detection-task
7. https://docs.aws.amazon.com/AmazonS3/latest/userguide/notification-content-structure.html

მადლობა ყურადღებისთვის!

კონტაქტი

თბილისი 0162, საქართველო
ი.ჭავჭავაძის გამზირი 82
INFO@BTU.EDU.GE
032 2 195 015
WWW.BTU.EDU.GE

გამოგვყევი:



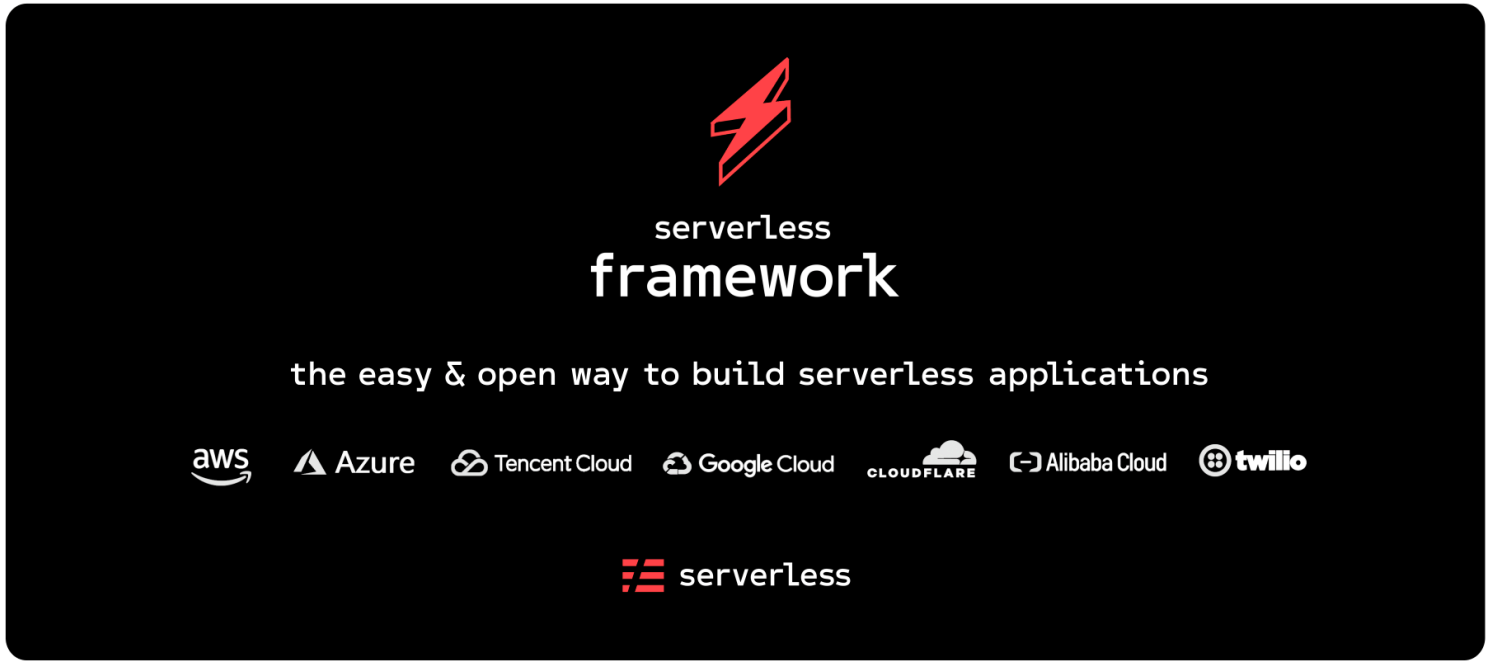
ღრუბლოვანი სისტემების (AWS) ავტომატიზაცია Python-ის საშუალებით

Guja Nemsadze

საკითხები

- Serverless Framework.
- Project 1 - Server-less Web Application.
- Project 2 - Analyse videos/images with AWS Recognition.
- AWS Lambda Anti-patterns

Serverless Framework

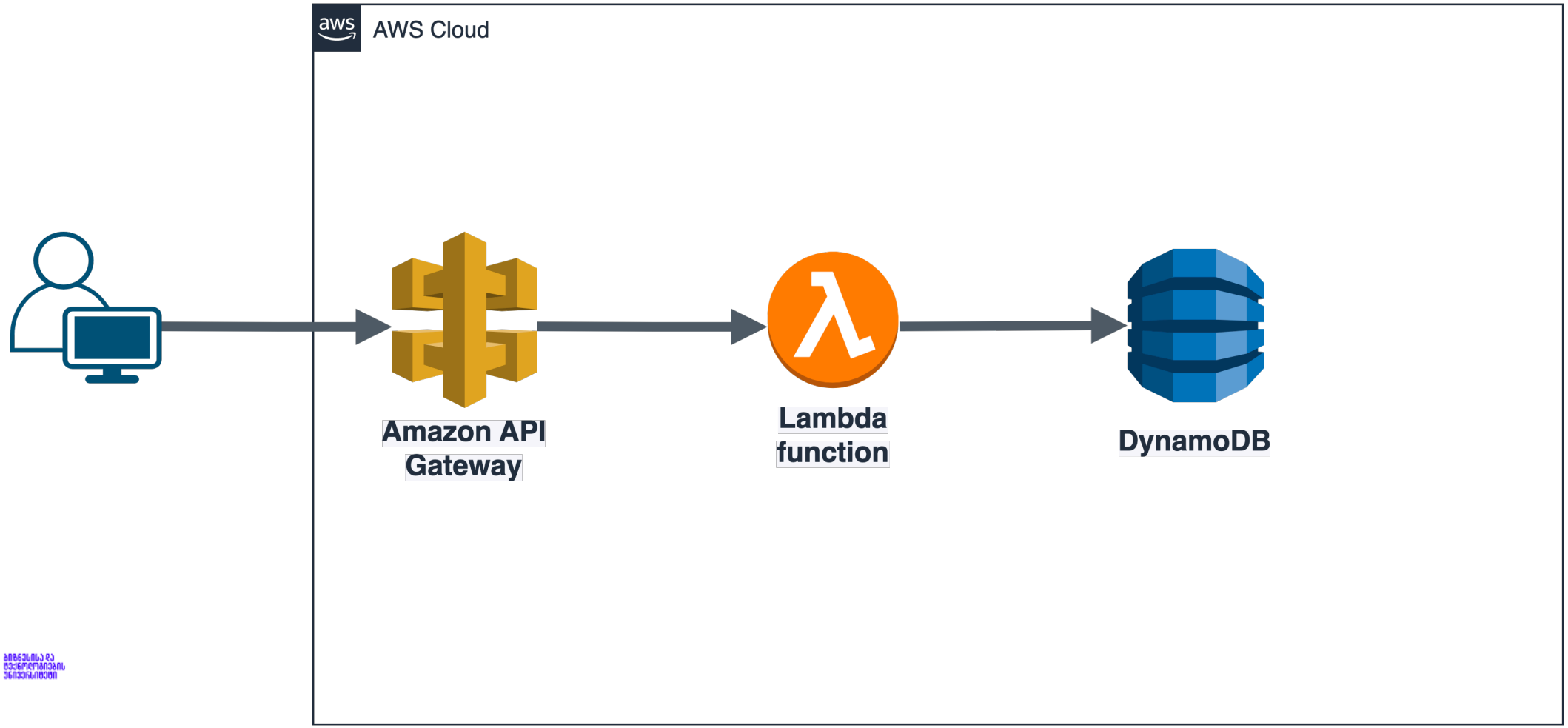


Under the hood, the Server-less Framework is deploying your code to a cloud provider like AWS, Microsoft Azure, Google Cloud Platform, Apache OpenWhisk, Cloudflare Workers, or a Kubernetes-based solution like Kubeless. Click on your provider for detailed Server-less Framework CLI user guides and references.

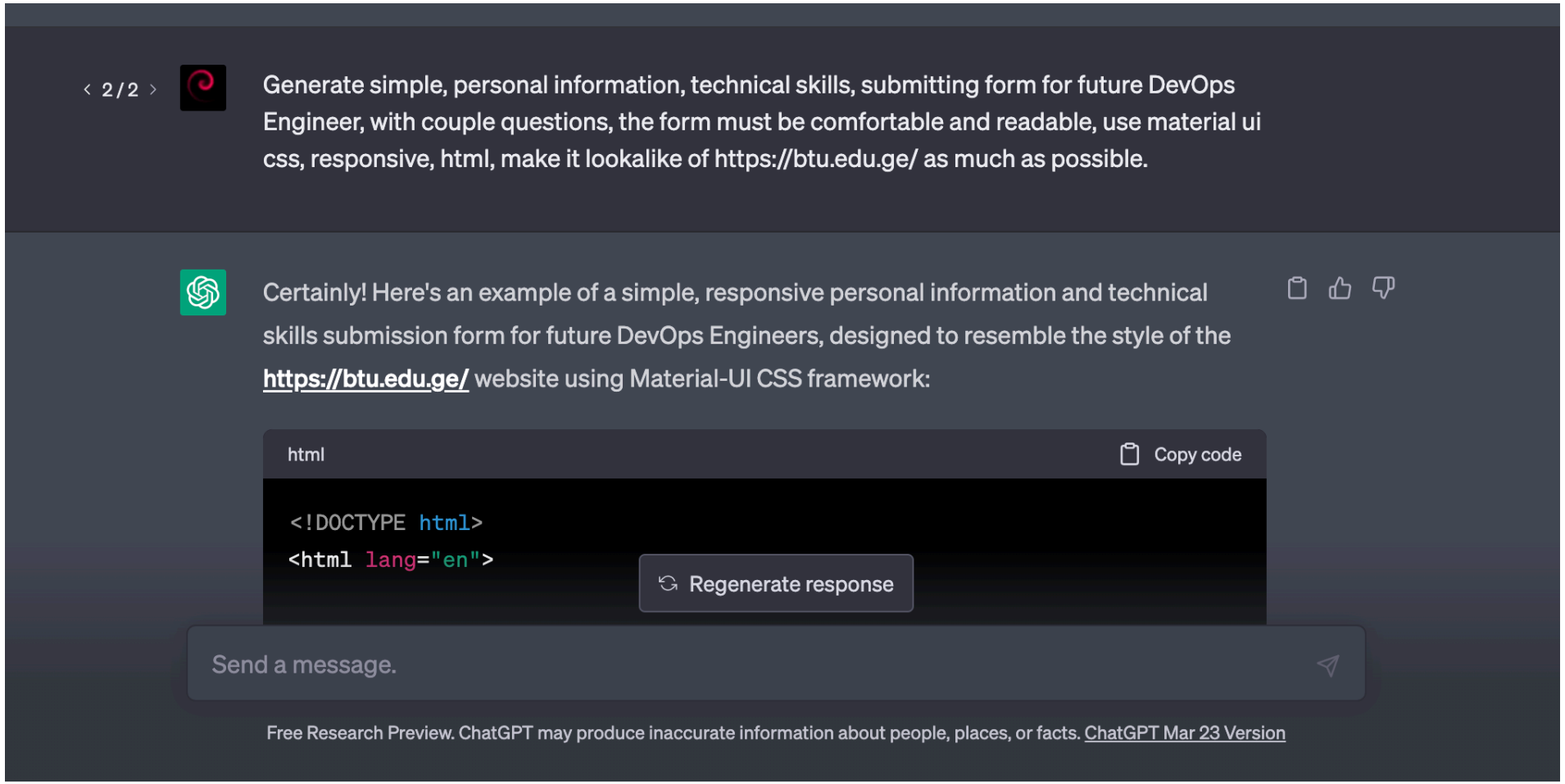
Serverless Frameworks for 2023.

Project 1

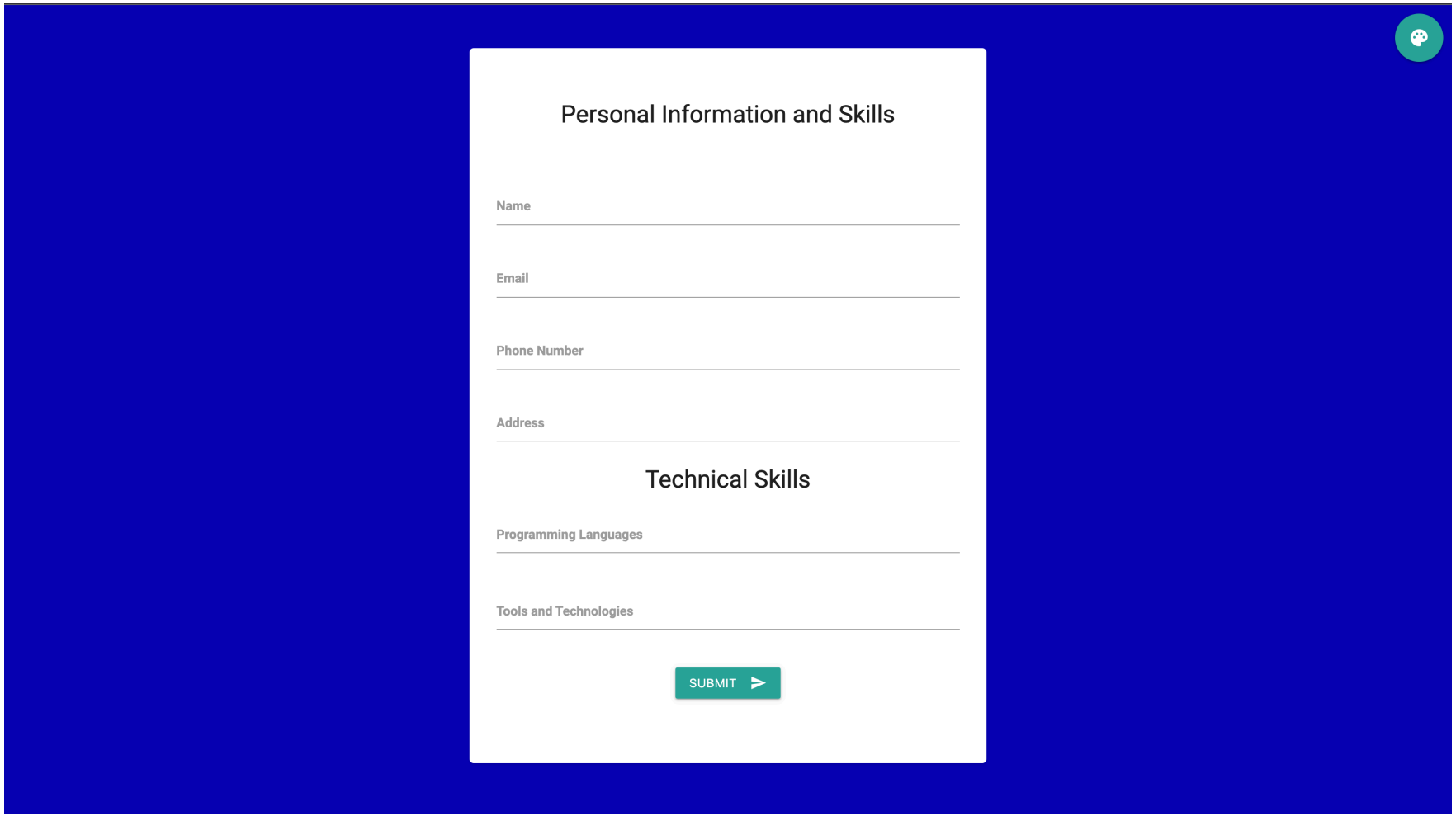
Building a Server-less Form with API Gateway.



Project 1


‹ 2 / 2 ›
Generate simple, personal information, technical skills, submitting form for future DevOps Engineer, with couple questions, the form must be comfortable and readable, use material ui css, responsive, html, make it lookalike of https://btu.edu.ge/ as much as possible.
Certainly! Here's an example of a simple, responsive personal information and technical skills submission form for future DevOps Engineers, designed to resemble the style of the https://btu.edu.ge/ website using Material-UI CSS framework:
html
Copy code
<!DOCTYPE html>
<html lang="en">
Regenerate response
Send a message.
Free Research Preview. ChatGPT may produce inaccurate information about people, places, or facts. ChatGPT Mar 23 Version

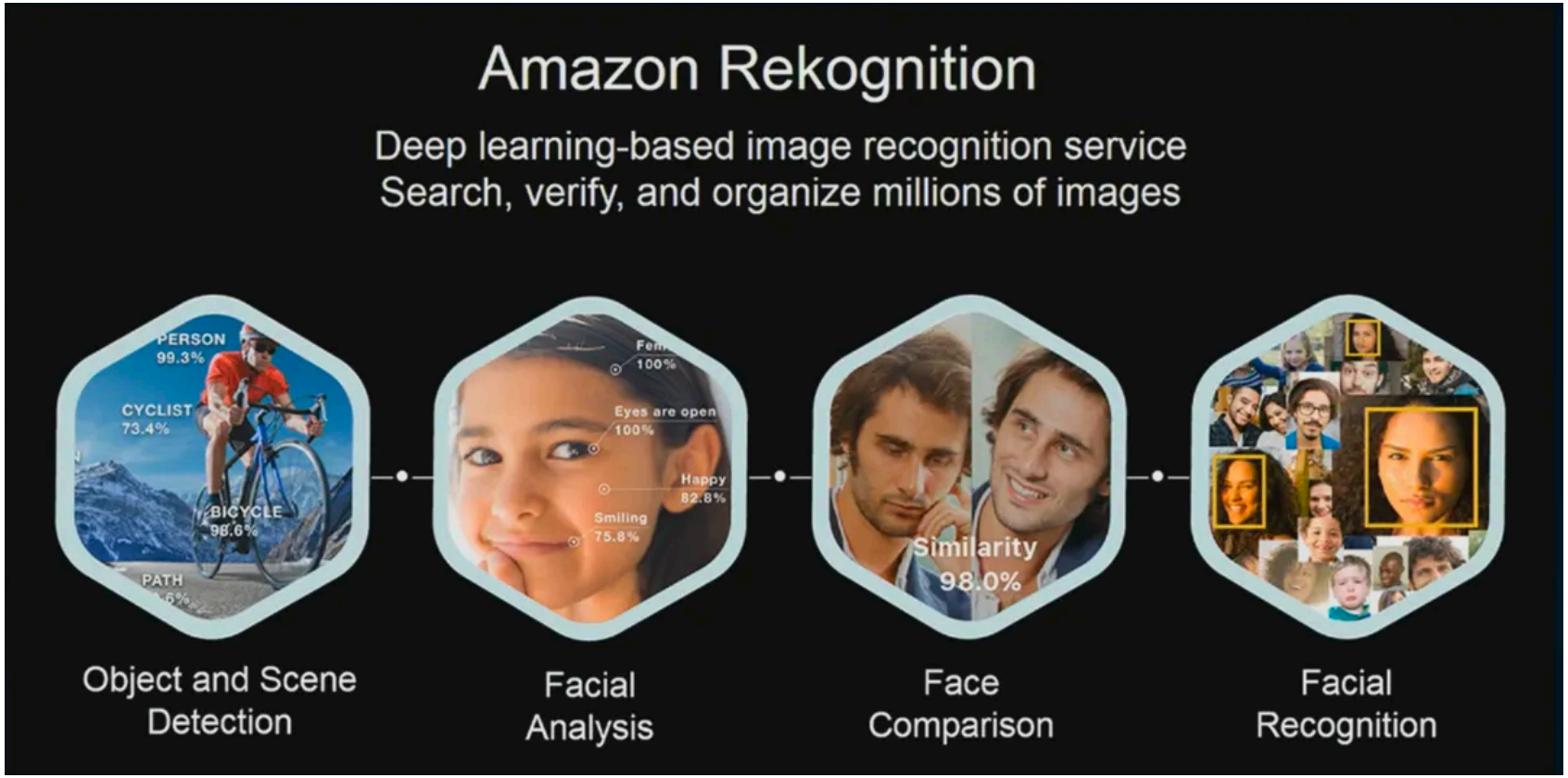
Project 1



https://replit.com/@JexPY/MonthlyHeftyArray

Project 2

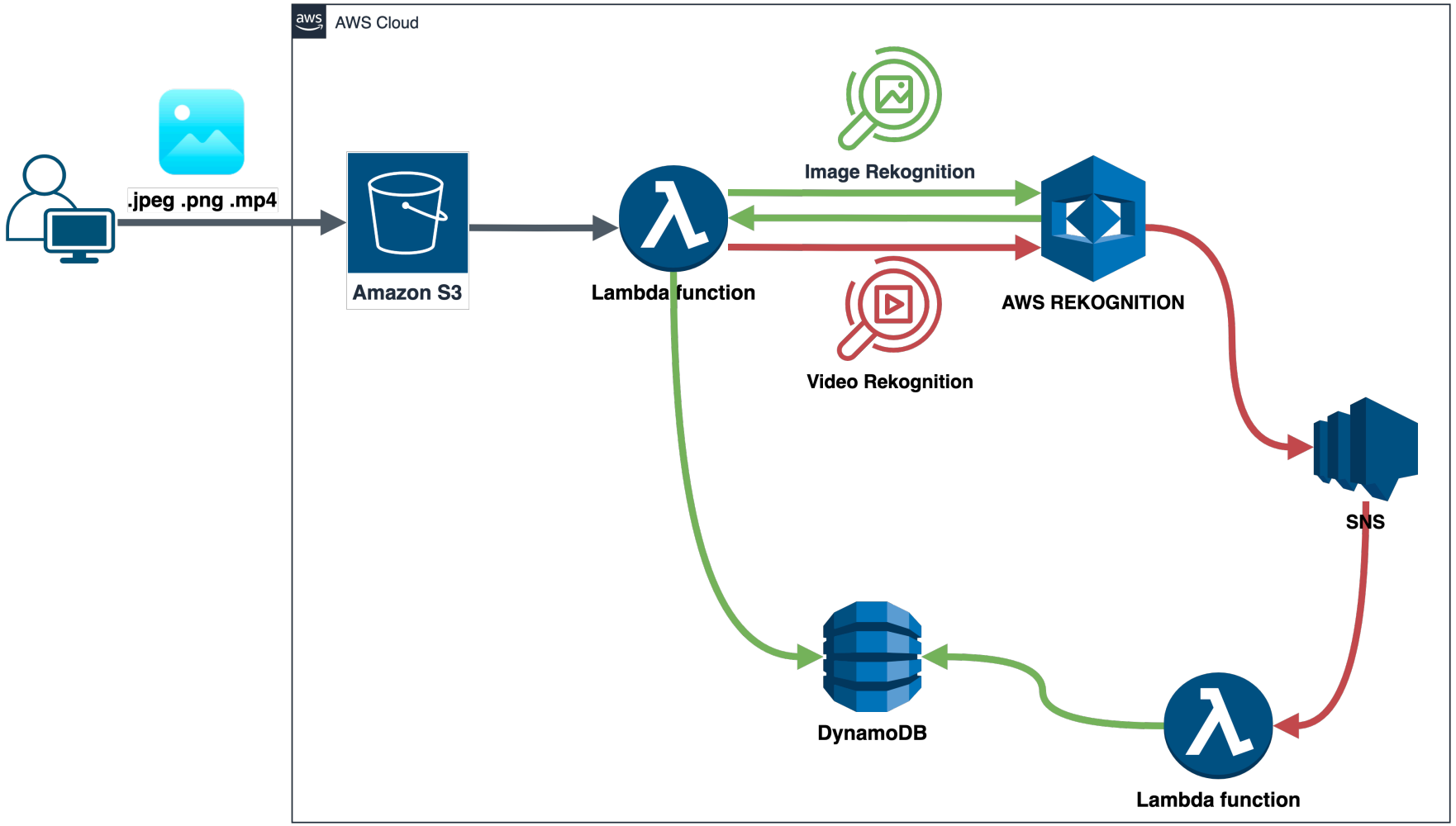
[AWS Rekognition: Cloud-based image and video analysis service using machine learning for object, face, text recognition, and more.]



Free license media: https://pixabay.com/videos/

Project 2

Building a service to analyse videos/images with AWS Recognition



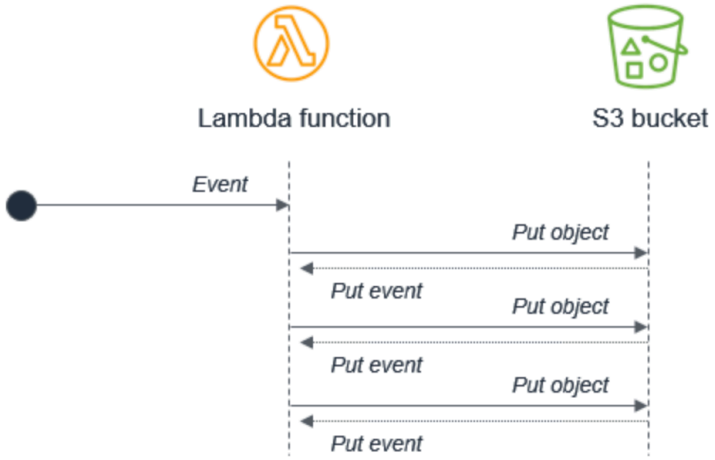
https://replit.com/@JexPY/PurpleBoringLinuxkernel

AWS Lambda Anti-patterns

Recursive patterns that cause run-away Lambda functions

AWS services generate events that invoke Lambda functions, and Lambda functions can send messages to AWS services. Generally, the service or resource that invokes a Lambda function should be different to the service or resource that the function outputs to. Failure to manage this can result in infinite loops.

For example, a Lambda function writes an object to an S3 object, which in turn invokes the same Lambda function via a put event. The invocation causes a second object to be written to the bucket, which invokes the same Lambda function:



Warning

If your Lambda function uses the same bucket that triggers it, it could cause the function to run in a loop. For example, if the bucket triggers a function each time an object is uploaded, and the function uploads an object to the bucket, then the function indirectly triggers itself. To avoid this, use two buckets, or configure the trigger to only apply to a prefix used for incoming objects.

https://docs.aws.amazon.com/lambda/latest/operatorguide/recursive-runaway.html

More: https://docs.aws.amazon.com/lambda/latest/operatorguide/anti-patterns.html

https://aws.amazon.com/blogs/compute/avoiding-recursive-invocation-with-amazon-s3-and-aws-lambda/

ლიტერატურა

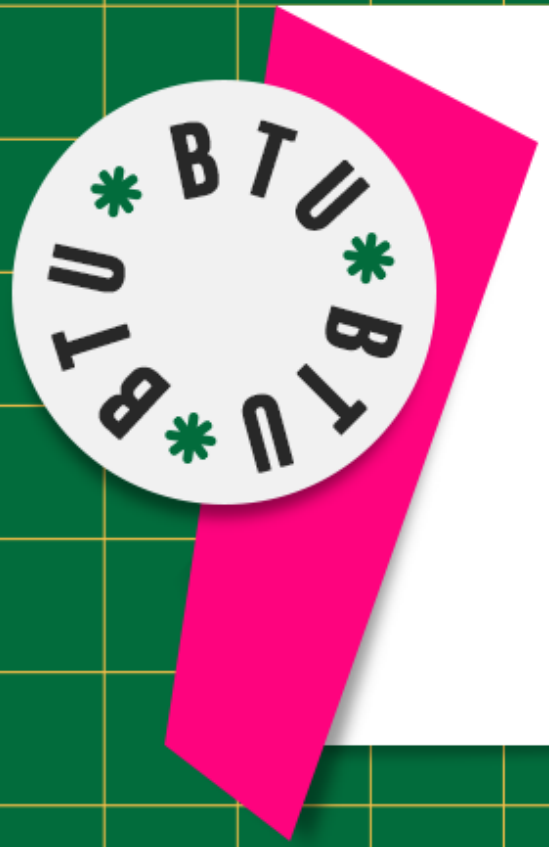
1. AWS Scripted: How to Automate the Deployment of Secure and Resilient Websites with Amazon Web Services VPC, ELB, EC2, RDS, IAM, SES and SNS - კრისტიან ჩერი, 2014წ. , 270 გვერდი
2. Amazon Web Services in Action - ანდრეას ვიტიგი, მიქაელ ვიტიგი, 2016წ., 424 გვ,
3. https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/ec2.html

მადლობა ყურადღებისთვის!

კონტაქტი

თბილისი 0162, საქართველო
ი.ჭავჭავაძის გამზირი 82
INFO@BTU.EDU.GE
032 2 195 015
WWW.BTU.EDU.GE

გამოგვყევით:



ღრუბლოვანი სისტემების (AWS) ავტომატიზაცია Python-ის საშუალებით

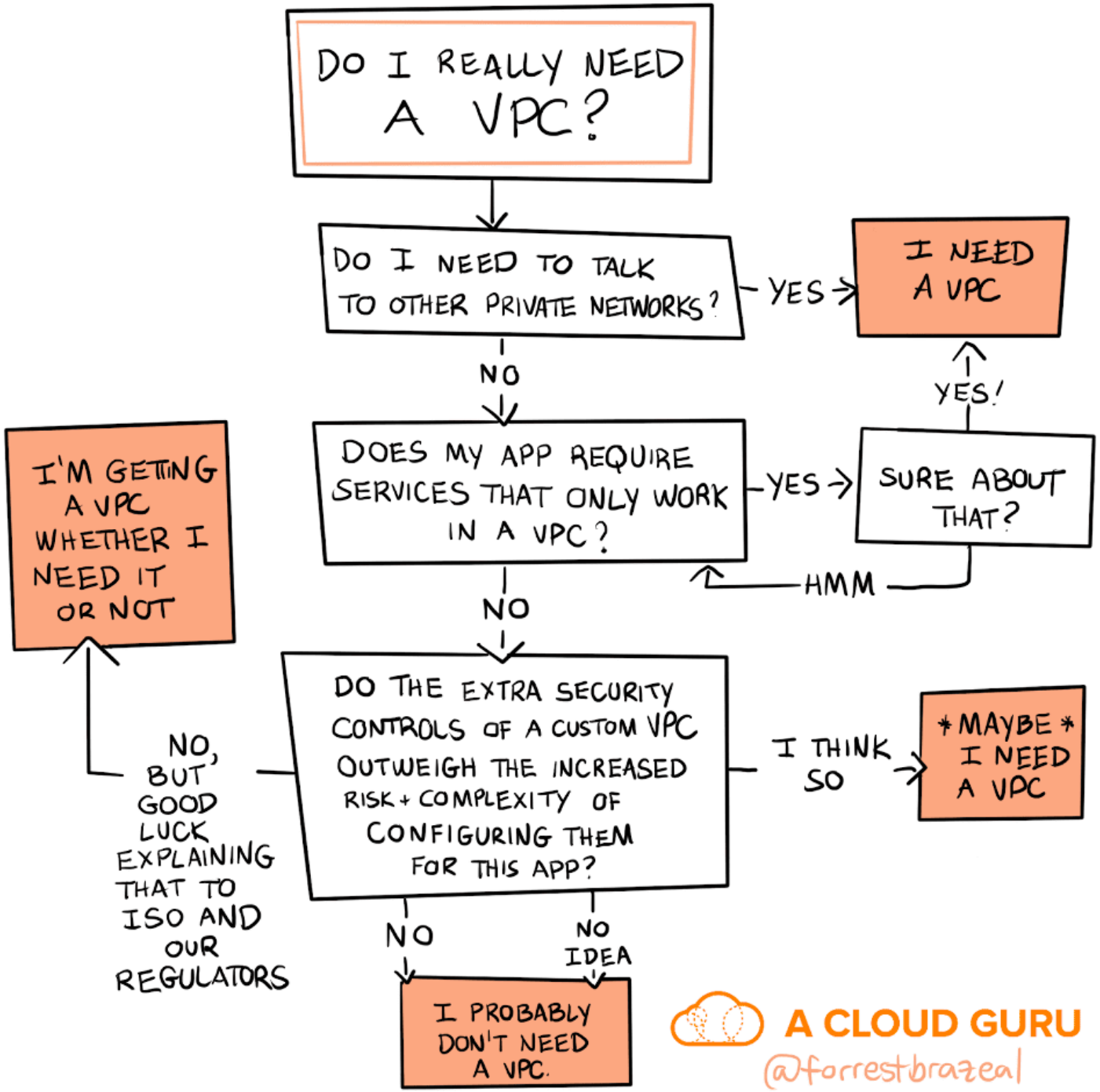
Guja Nemsadze

საკითხები

- VPC - Virtual Private Cloud
- IP addressing for your VPCs and subnets
- Do I really need VPC?
- VPC
- Every network-connected device must have a unique IP address!
- What is Network Address Translation (NAT)?
- NAT allows for very sophisticated network segmenting.
- Defining Routing Protocols
- Difference between Internet Gateway and NAT Gateway
- Run Python Code

Why should I use Amazon VPC?

Amazon VPC enables you to build a virtual network in the AWS cloud - no VPNs, hardware, or physical datacenter required. You can define your own network space, and control how your network and the Amazon EC2 resources inside your network are exposed to the Internet. You can also leverage the enhanced security options in Amazon VPC to provide more granular access to and from the Amazon EC2 instances in your virtual network.

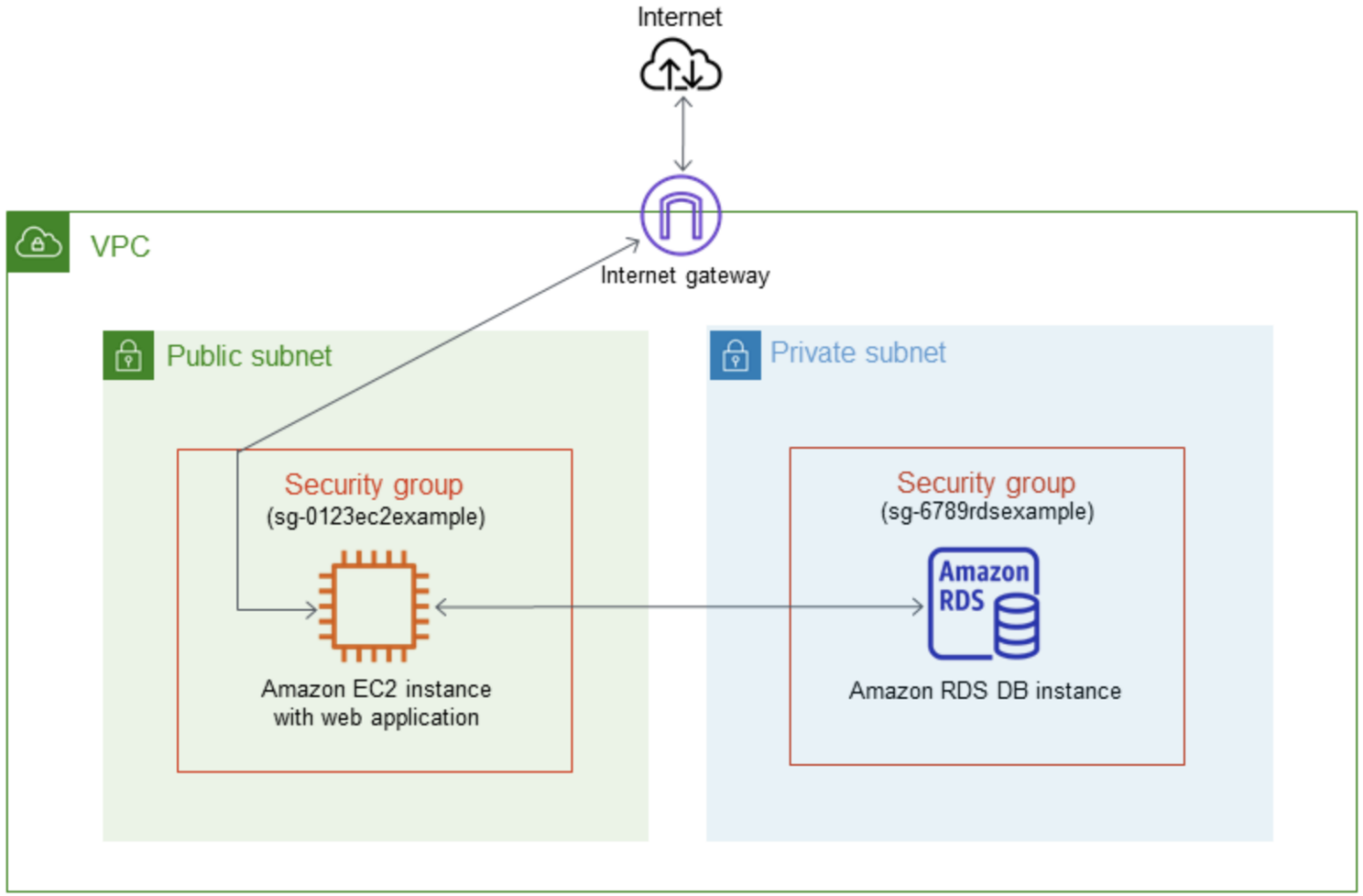

DO I REALLY NEED A VPC?
DO I NEED TO TALK TO OTHER PRIVATE NETWORKS?
YES
I NEED A VPC
NO
YES!
DOES MY APP REQUIRE SERVICES THAT ONLY WORK IN A VPC?
YES
SURE ABOUT THAT?
HMM
NO
I'M GETTING A VPC WHETHER I NEED IT OR NOT
DO THE EXTRA SECURITY CONTROLS OF A CUSTOM VPC OUTWEIGH THE INCREASED RISK + COMPLEXITY OF CONFIGURING THEM FOR THIS APP?
NO, BUT GOOD LUCK EXPLAINING THAT TO ISO AND OUR REGULATORS
I THINK SO
MAYBE I NEED A VPC
NO
NO IDEA
I PROBABLY DON'T NEED A VPC.
A CLOUD GURU
@forrestbrazeal

VPC

Amazon Virtual Private Cloud is a service that lets you launch AWS resources in a **logically isolated virtual network** that you define.

You get complete control over the networking environment including IP address ranges, subnets, routing, firewalls and more.

From a security standpoint, a VPC isn't a magic power. It's another layer of responsibility.



*"**The VPC** doesn't do anything, really, You need a proper network architecture with NACLs (network access control list) , subnets, and security groups. You need to know how to build the architecture so you can monitor for attacks. People need to understand network layers, attacks, and how attackers pivot through networks."*

https://acloudguru.com/blog/engineering/do-i-really-need-a-vpc

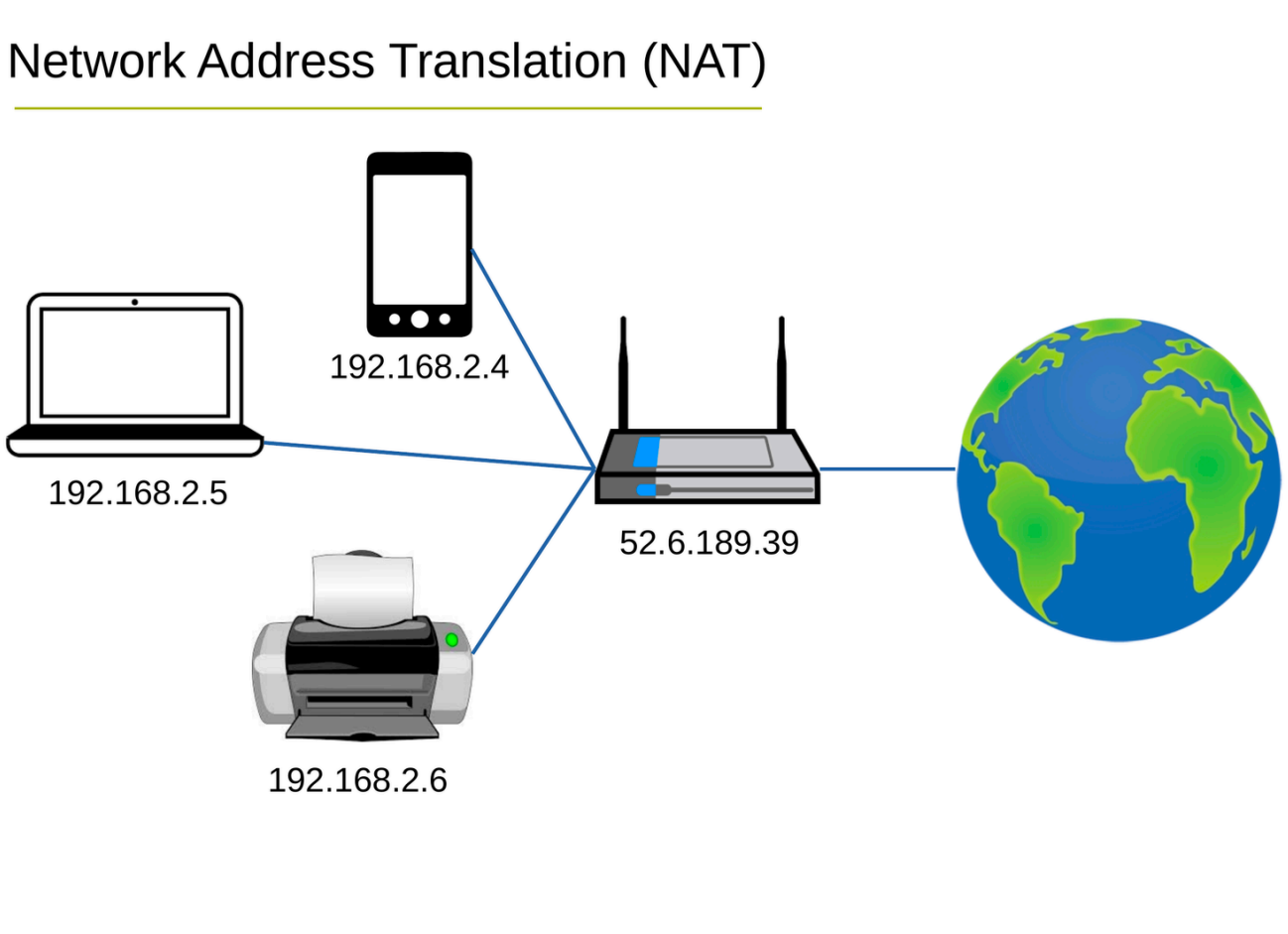
IP addressing for your VPCs and subnets

- **IP** is responsible for routing data packets from one device to another.
- **IP version 4** is the most widely used IP address format and consists of four 32-bit numbers separated by dots. (Example: 192.168.2.34)
- **IP version 6** is the latest IP address format, which uses eight 128-bit numbers separated by colons. (Example: 2001:0db8:85a3:0000:0000:8a2e:0370:7334)

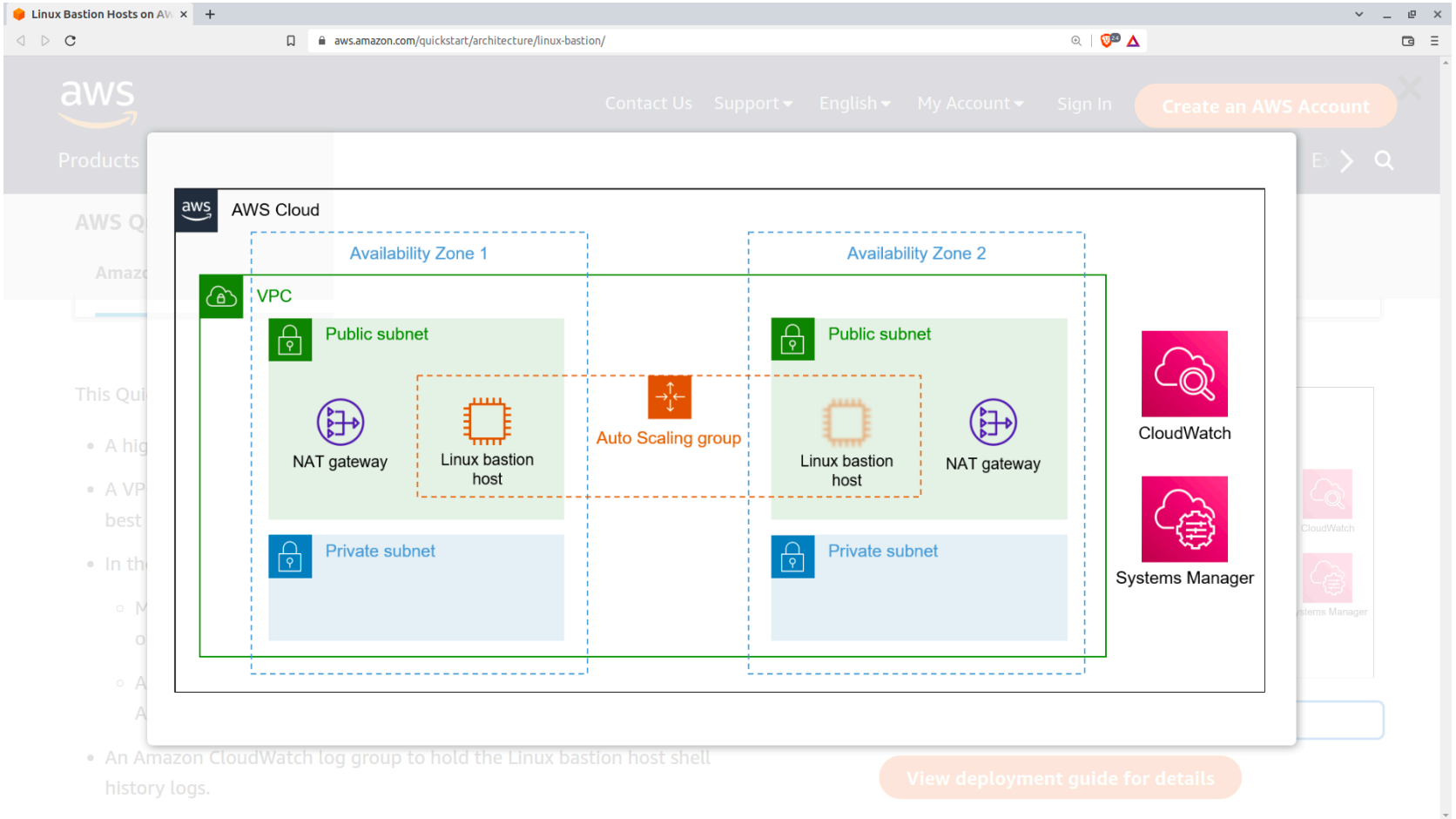
Every network-connected device must have a unique IP address!

- Mathematically, there can be no more than four billion 32-bit **IPv4** addresses, and that there are already far more than four billion network-connected devices on the internet, something had to change.
- The 128-bit **IPv6** protocol was eventually introduced to allow trillions of unique addresses. **We'll never run out of those.**
- But before IPv6, another brilliant solution was introduced: **NAT networking**.

What is Network Address Translation (NAT)?



NAT allows for very sophisticated network segmenting.

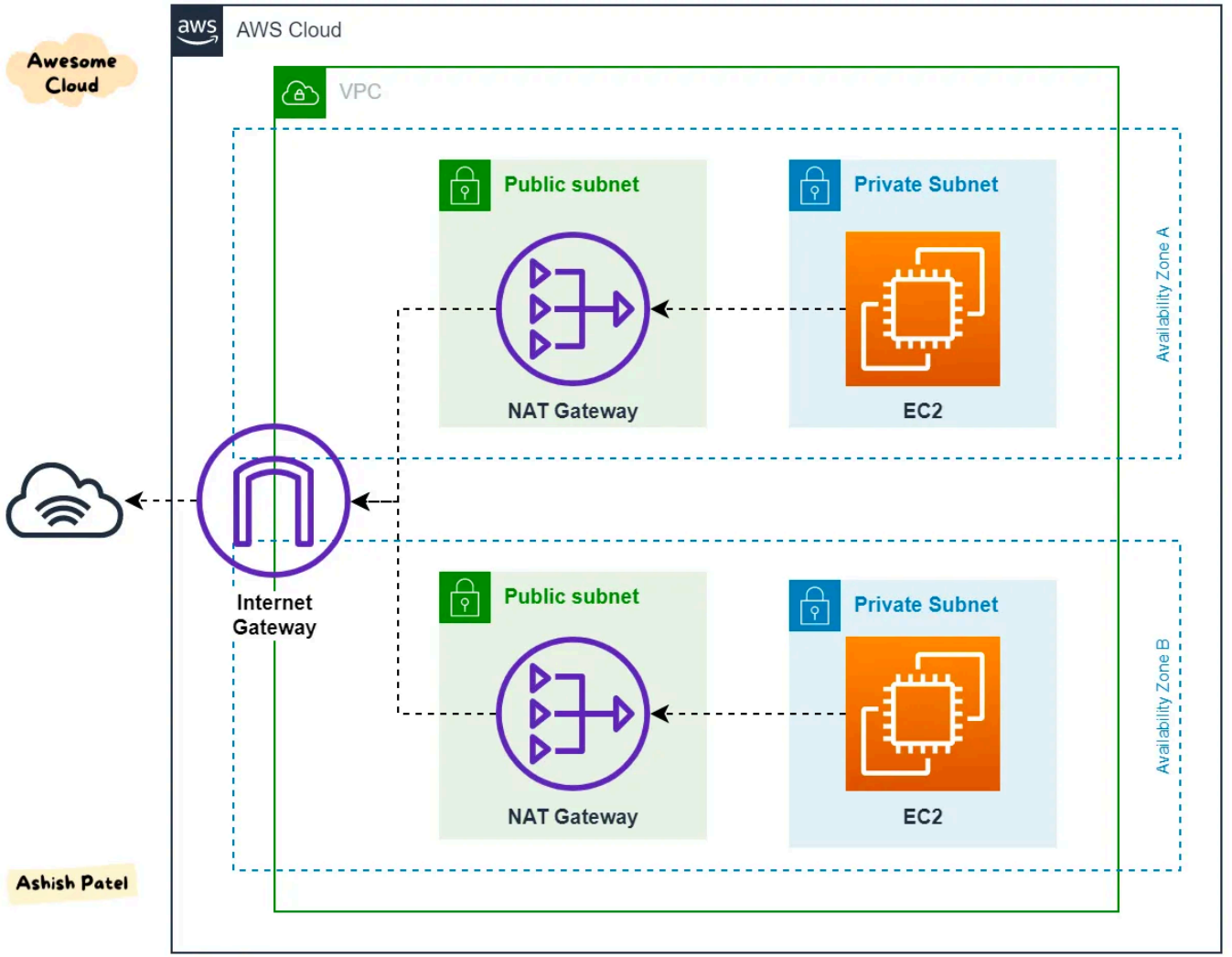


The bastion host provides a jump box allowing admins to safely open remote SSH sessions on instances running in your private subnets. And the NAT gateway allows services running on your private instances outbound access to, for example, pull software updates. Both bastion hosts and NAT gateways will incur regular usage costs, by the way.

Defining Routing Protocols

- TCP:
  - HTTP (Hypertext Transfer Protocol) - used for web browsing
  - FTP (File Transfer Protocol) - used for file transfers
  - SMTP (Simple Mail Transfer Protocol) - used for sending email
  - SSH (Secure Shell) - used for secure remote access
  - Telnet - used for remote access
- UDP:
  - DNS (Domain Name System) - used for translating domain names into IP addresses
  - DHCP (Dynamic Host Configuration Protocol) - used for dynamically assigning IP addresses
  - VoIP (Voice over Internet Protocol) - used for real-time audio and video communication
  - Online gaming - used for fast data transmission to reduce lag
- ICMP:
  - Ping - used to test network connectivity and measure latency
  - Traceroute - used to identify the path that packets take from one host to another
  - Path MTU Discovery - used to determine the maximum transmission unit (MTU) size for a network path
- DHCP:
  - Home and small business networks - used for dynamically assigning IP addresses to devices on the network

Difference between Internet Gateway and NAT Gateway



- **Internet Gateway (IGW)** allows instances with public IPs to access the internet.
- **NAT Gateway (NGW)** allows instances with no public IPs to access the internet.

Run Python Code To

- list vpcs
- create_vpc
- add_name_tag
- create_igw
- attach_igw_to_vpc

https://replit.com/@JexPY/EachOtherMonitor#main.py

ლიტერატურა

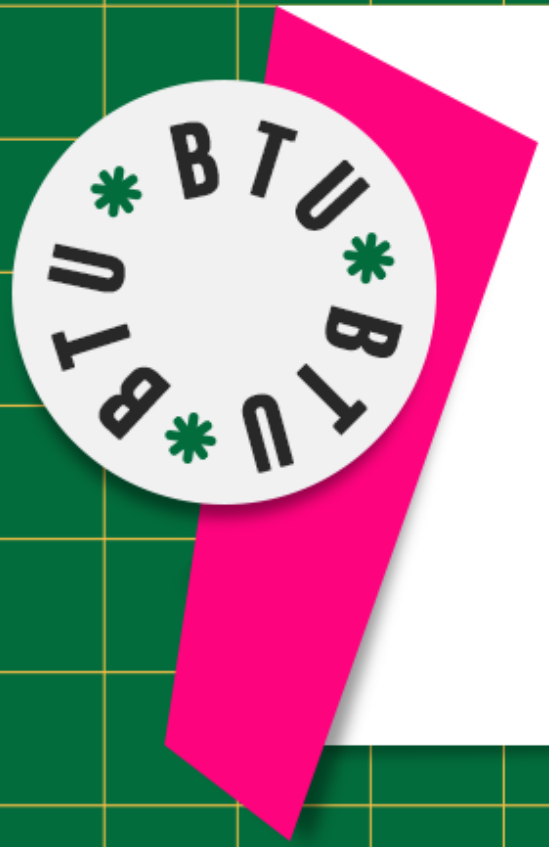
1. AWS Scripted: How to Automate the Deployment of Secure and Resilient Websites with Amazon Web Services VPC, ELB, EC2, RDS, IAM, SES and SNS - კრისტიან ჩერი, 2014წ. , 270 გვერდი
2. Amazon Web Services in Action - ანდრეას ვიტიგი, მიქაელ ვიტიგი, 2016წ., 424 გვ,
3. https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/ec2.html

მადლობა ყურადღებისთვის!

კონტაქტი

თბილისი 0162, საქართველო
ი.ჭავჭავაძის გამზირი 82
INFO@BTU.EDU.GE
032 2 195 015
WWW.BTU.EDU.GE

გამოგვყევი:



ღრუბლოვანი სისტემების (AWS) ავტომატიზაცია Python-ის საშუალებით

Guja Nemsadze

საკითხები

- IPv4 Address Composition
- Classful Networking
- CIDR
- AWS VPC rules
- Private/Public Subnet with Boto3
- Bonus Code (S3 -> Lambda -> HugginFaceModel -> Dynamodb)

IPv4 Address Composition

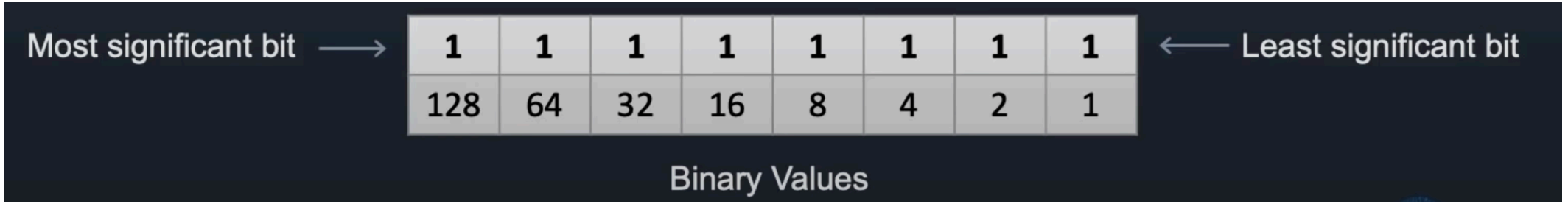
172.16.0.0

10101100.00010000.00000000.00000000

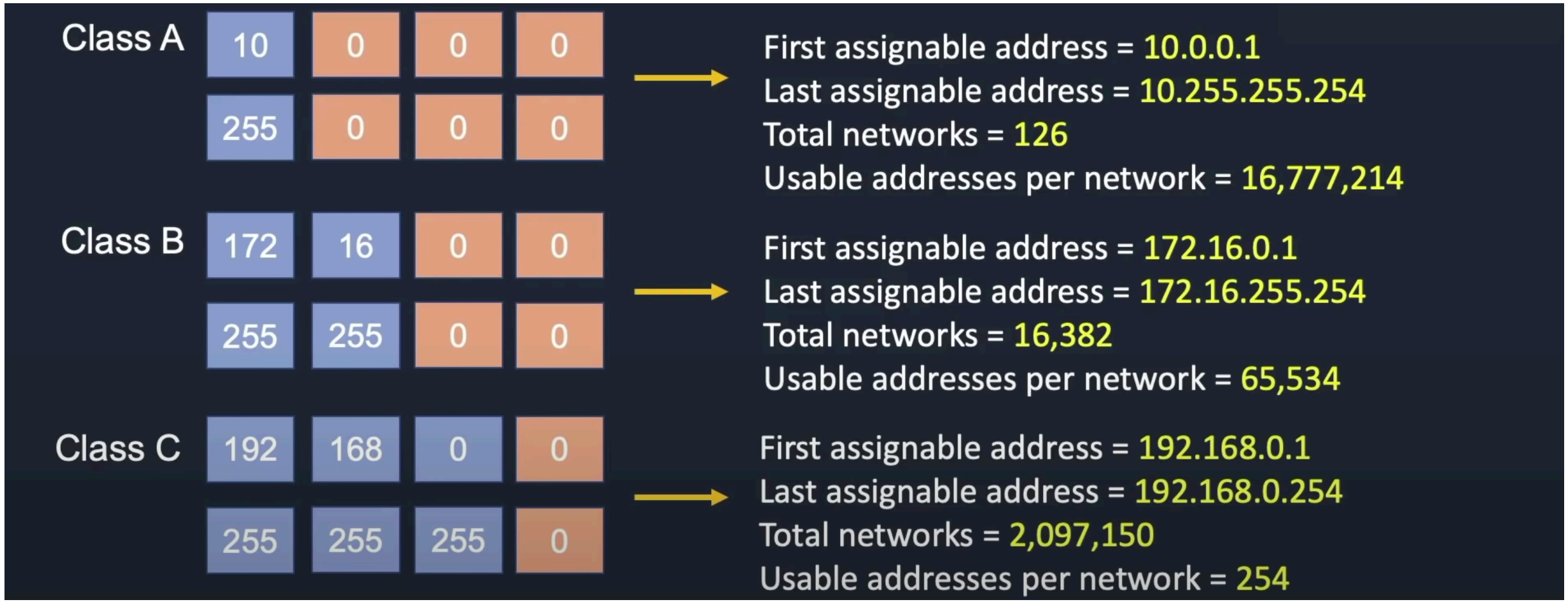
Octet #1 ..... Octet #4

IPv4 Address Composition

- 32 bit value
- 2^32 total address possibilities = 4,294,967,296



Classful Networking



If an organization needed more than 254 host machines, it would be switched into Class B. However, this could potentially waste over 60,000 hosts if the business didn't need to use them, thus unnecessarily decreasing the availability of IPv4 addresses. The Internet Engineering Task Force introduced **CIDR** in 1993 to fix this problem.

CIDR (Classless Inter-Domain Routing)

Classless Inter-Domain Routing (CIDR) is a range of IP addresses a network uses. A CIDR address looks like a normal IP address, except that it ends with a slash followed by a number. The number after the slash represents the number of addresses in the range.

CIDR Example:

In *IPv4*:
192.0.2.0/24
Because IPv4 has a 32-bit address space, the 24-bit prefix in the preceding example means that the address range is the 8 bits (256 addresses) after 192.0.2.0.

In *IPv6*:
2001:db8::/32
IPv6 has a 128-bit address range, so the 32-bit network prefix refers to 96 bits worth of addresses following 2001:db8::, about 79 octillion addresses.

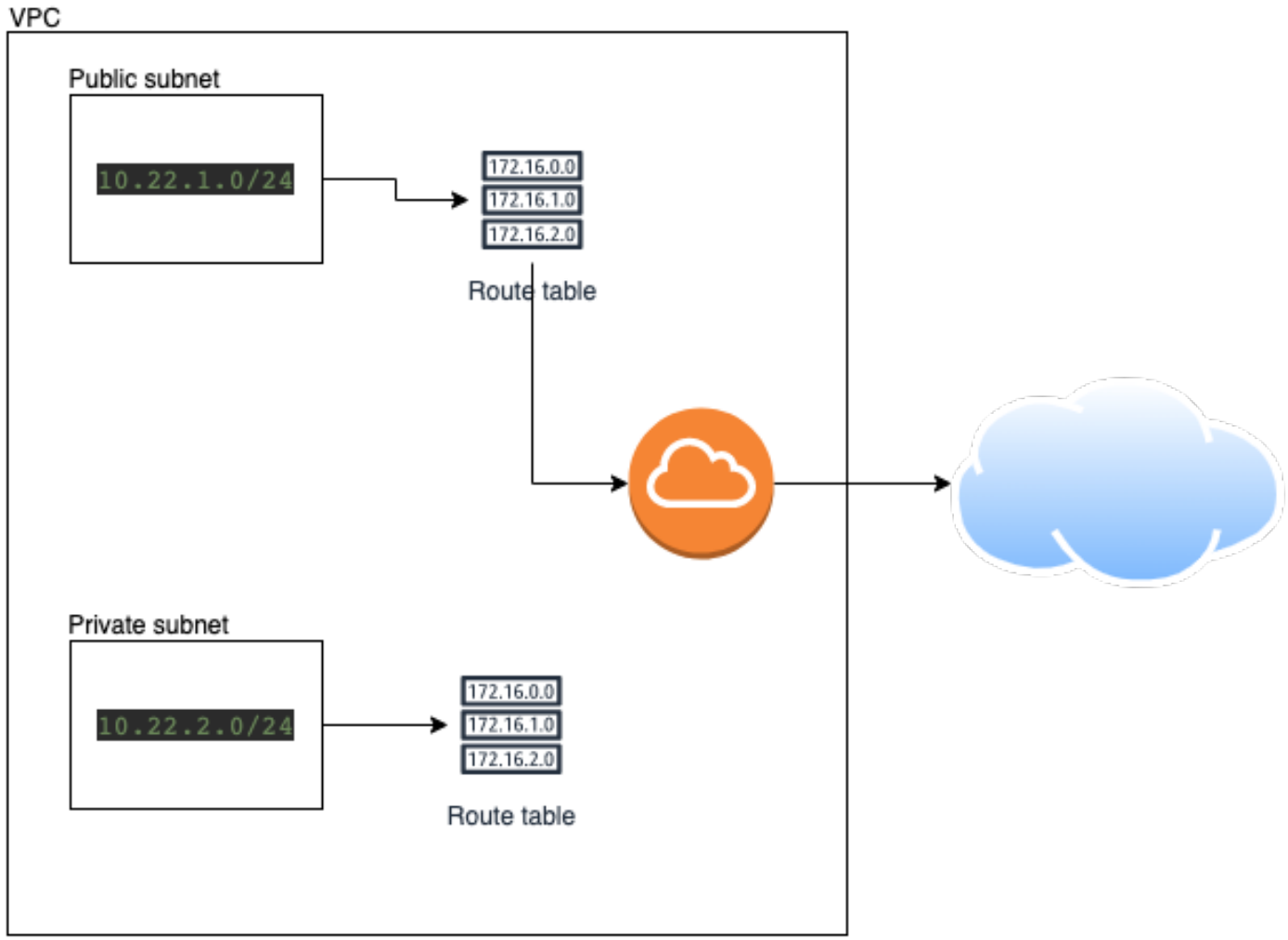
AWS VPC rules

- CIDR block size can be between /16 and /28
- The CIDR block must not overlap with any existing CIDR block that's associated with the VPC
- You cannot increase or decrease the size of an existing CIDR block
- The first four and last IP address are not available for use
- AWS recommend you use CIDR blocks from the RFC 1918 ranges:

RFC 1918 Range	Example CIDR Block
10.0.0.0 - 10.255.255.255 (10/8 prefix)	Your VPC must be /16 or smaller, for example, 10.0.0.0/16
172.16.0.0 - 172.31.255.255 (172.16/12 prefix)	Your VPC must be /16 or smaller, for example, 172.31.0.0/16
192.168.0.0 - 192.168.255.255 (192.168/16 prefix)	Your VPC can be smaller, for example 192.168.0.0/20

https://docs.aws.amazon.com/vpc/latest/userguide/vpc-cidr-blocks.html

დიაგრამა



https://www.site24x7.com/tools/ipv4-subnetcalculator.html

Private/Public Subnet with Boto3

https://replit.com/@JexPY/ImmenseShamefulDatabases#main.py

Bonus Code

(S3 -> Lambda -> HugginFaceModel -> Dynamodb)

https://replit.com/@JexPY/DifferentQuickStructs#main.py

ლიტერატურა

1. AWS Scripted: How to Automate the Deployment of Secure and Resilient Websites with Amazon Web Services VPC, ELB, EC2, RDS, IAM, SES and SNS - კრისტიან ჩერი, 2014წ. , 270 გვერდი
2. Amazon Web Services in Action - ანდრეას ვიტიგი, მიქაელ ვიტიგი, 2016წ., 424 გვ,
3. https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/ec2.html

მადლობა ყურადღებისთვის!

კონტაქტი

თბილისი 0162, საქართველო
ი.ჭავჭავაძის გამზირი 82
INFO@BTU.EDU.GE
032 2 195 015
WWW.BTU.EDU.GE

გამოგვყევით:

ღრუბლოვანი სისტემების (AWS) ავტომატიზაცია Python-ის საშუალებით

საკითხები

- What is Amazon EC2?
- What is an AMI?
- What is an Instance Type?
- Amazon EC2 general-purpose instances
- Amazon EC2 memory-optimized instances
- Amazon EC2 storage-optimized instances
- EC2 UltraClusters
- Amazon EC2 purchase options
- Hands On EC2

What is Amazon EC2?

Amazon EC2 is the Amazon Elastic Compute Cloud (EC2) – is a Virtual Machine – which is nothing new. But what makes Amazon EC2 special, is what it can provide in terms of scalability, flexibility and raw compute power.

What is an AMI?

An Amazon Machine Image (AMI) is basically a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, the user community, or the AWS Marketplace; or you can select one of your own AMIs.

Amazon Linux

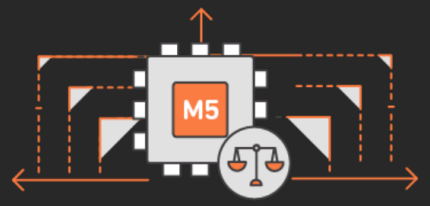
macOS

Red Hat

SUSE Linux

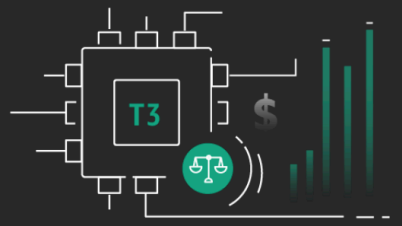
Windows

Amazon EC2 general-purpose instances



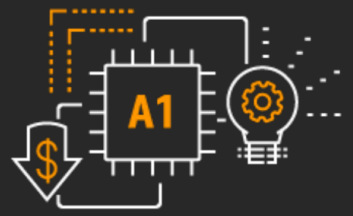
M5 instances

Balance of compute, memory, and network resources. 4:1 memory to vCPU ratio



T3 instances

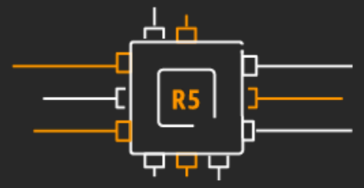
Baseline level of CPU performance with the ability to burst above the baseline for workloads that don't require sustained performance



A1 instances

Workloads that can scale out across multiple cores, fit within memory, run on ARM instructions

Amazon EC2 memory-optimized instances



R5 instances

Accelerate performance for workloads that process large data sets in memory
8:1 memory to vCPU ratio

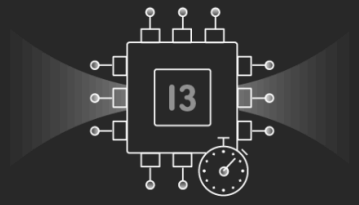
X1 / X1e instances

For memory-intensive workloads and very large in-memory workloads
16:1 and 32:1 memory to vCPU ratio

High memory instances

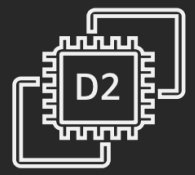
Extreme memory needs
Certified to run SAP HANA
From 6 to 24 TB of memory

Amazon EC2 storage-optimized instances



I3 / I3en instances

I/O optimized for high transaction workloads, low latency workloads



D2 instances

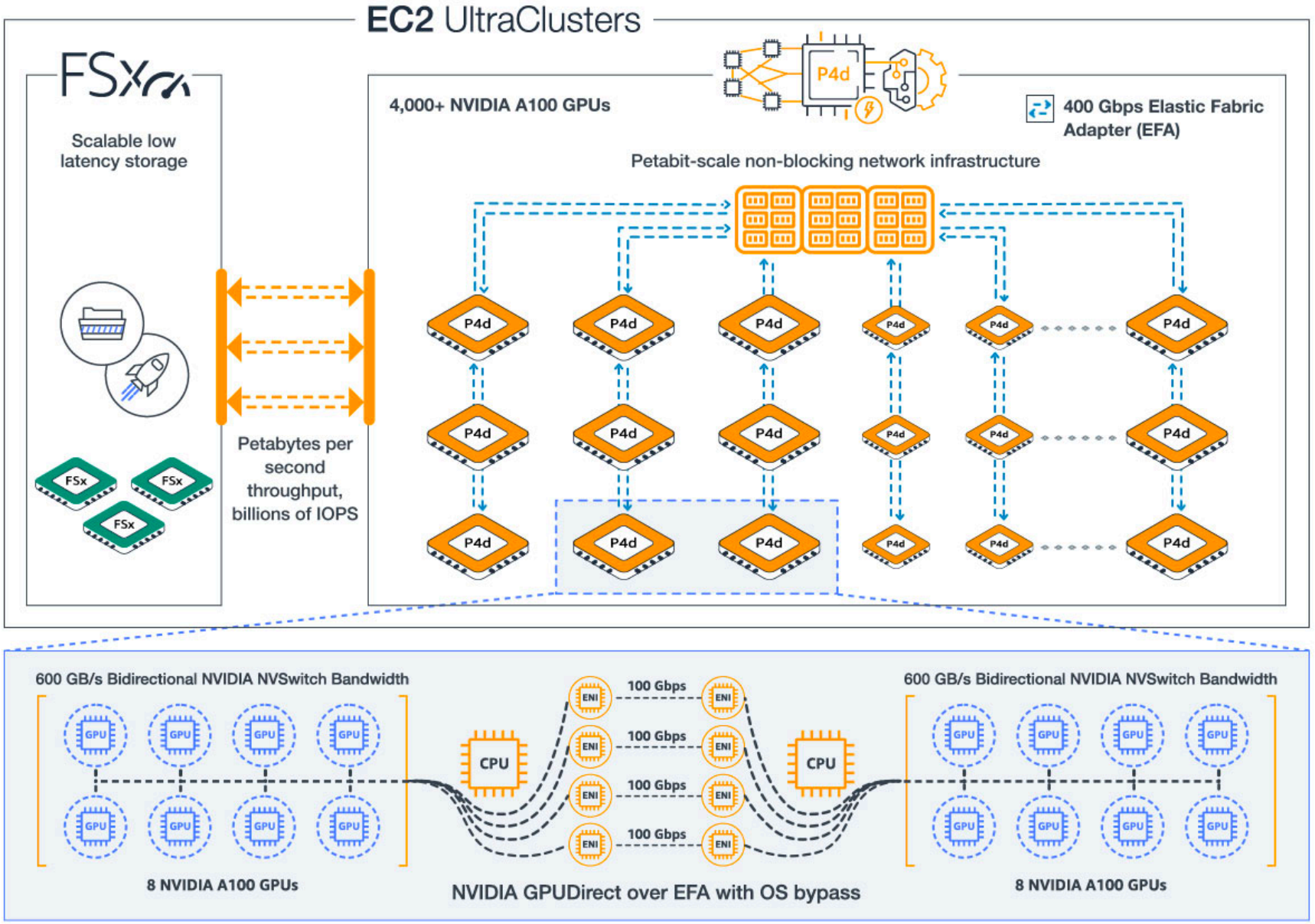
Lowest cost per storage ($/GB)

Supports high sequential disk throughput

H1 instances

Designed for applications that require low cost, high disk throughput and high sequential disk I/O access to very large data sets

More vCPUs and memory per TB of disk than D2

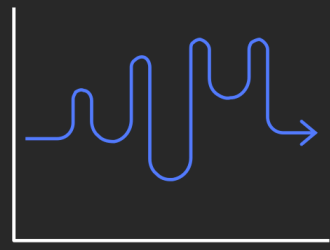


Any ML developer, researcher, or data scientist can spin up P4d instances in EC2 UltraClusters to get access to supercomputer-class performance with pay-as-you-go usage model to run their most complex multi-node ML training and HPC workloads.

Amazon EC2 purchase options

On-Demand

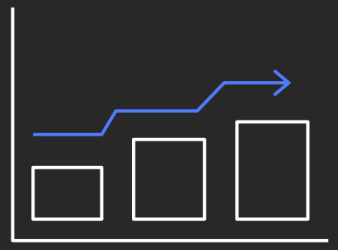
Pay for compute capacity by **the second** with no long-term commitments



Spiky workloads, to define needs

Reserved Instances

Make a 1- or 3-year commitment and receive a **significant discount** off On-Demand prices



Committed and steady-state usage

Savings Plan

Same great discounts as EC2 RIs with **more flexibility**

Flexibility to access compute across EC2 and AWS Fargate

NEW! Savings Plan

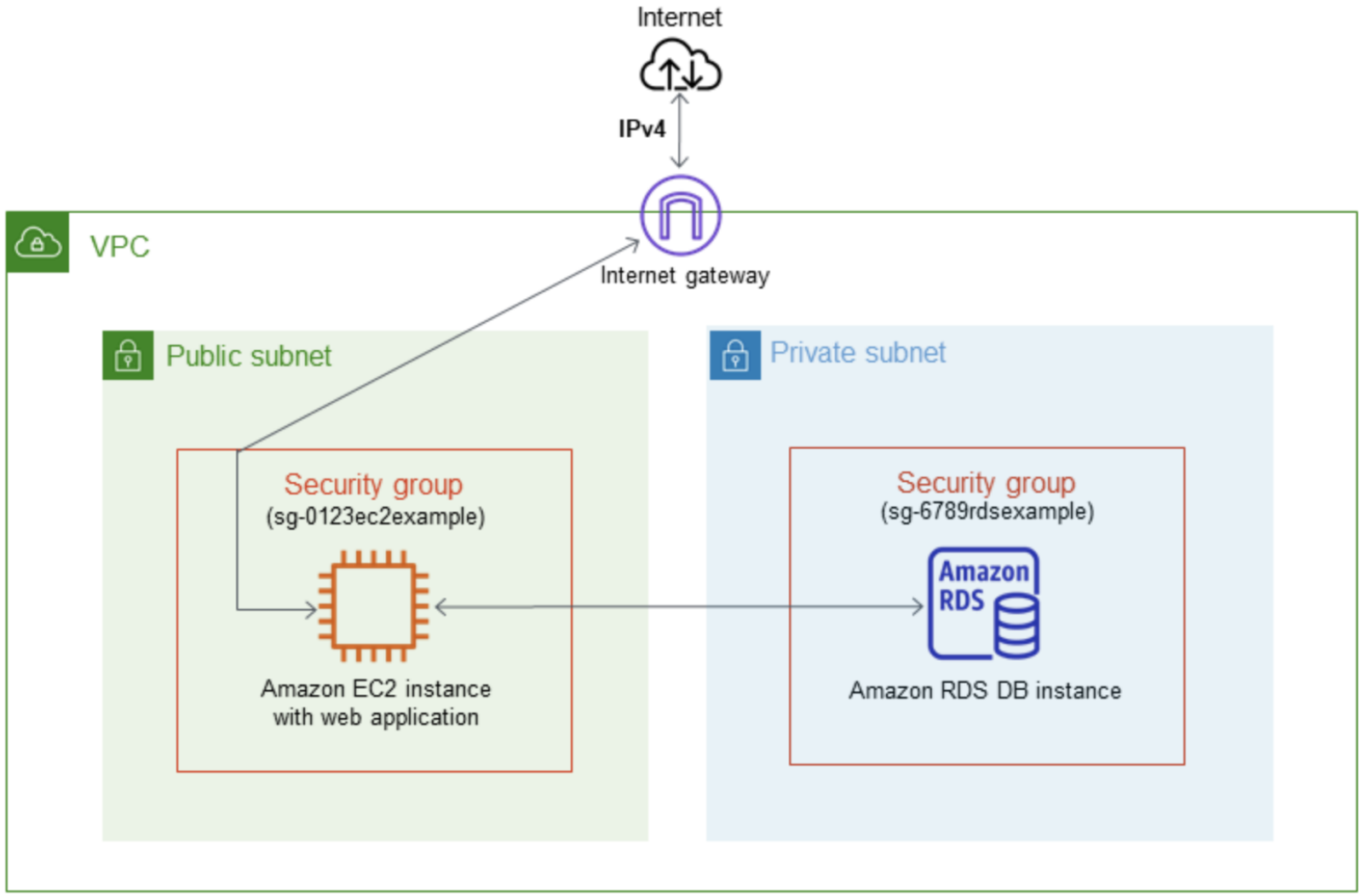
Spot Instances

Spare EC2 capacity at **savings of up to 90%** off On-Demand prices

Fault-tolerant, flexible, stateless workloads

[More to read...]

Hands On EC2


Internet
IPv4
Internet gateway
VPC
Public subnet
Private subnet
Security group
(sg-0123ec2example)
Amazon EC2 instance
with web application
Security group
(sg-6789rdsexample)
Amazon
RDS
Amazon RDS DB instance

Hands On EC2

- Create VPC or Use Already Created One
- Create Key-Pair (to connect with SSH)
- Create EC2

https://replit.com/@JexPY/DifficultMulticoloredSites

ლიტერატურა

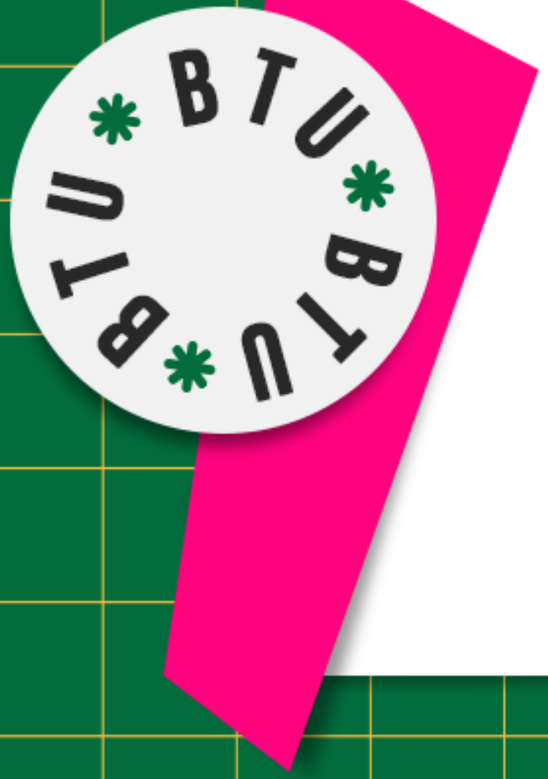
1. AWS Scripted: How to Automate the Deployment of Secure and Resilient Websites with Amazon Web Services VPC, ELB, EC2, RDS, IAM, SES and SNS - კრისტიან ჩერი, 2014წ. , 270 გვერდი
2. Amazon Web Services in Action - ანდრეას ვიტიგი, მიქაელ ვიტიგი, 2016წ., 424 გვ,
3. https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/ec2.html

მადლობა ყურადღებისთვის!

კონტაქტი

თბილისი 0162, საქართველო
ი.ჭავჭავაძის გამზირი 82
INFO@BTU.EDU.GE
032 2 195 015
WWW.BTU.EDU.GE

გამოგვყევით:



ღრუბლოვანი სისტემების (AWS) ავტომატიზაცია Python-ის საშუალებით

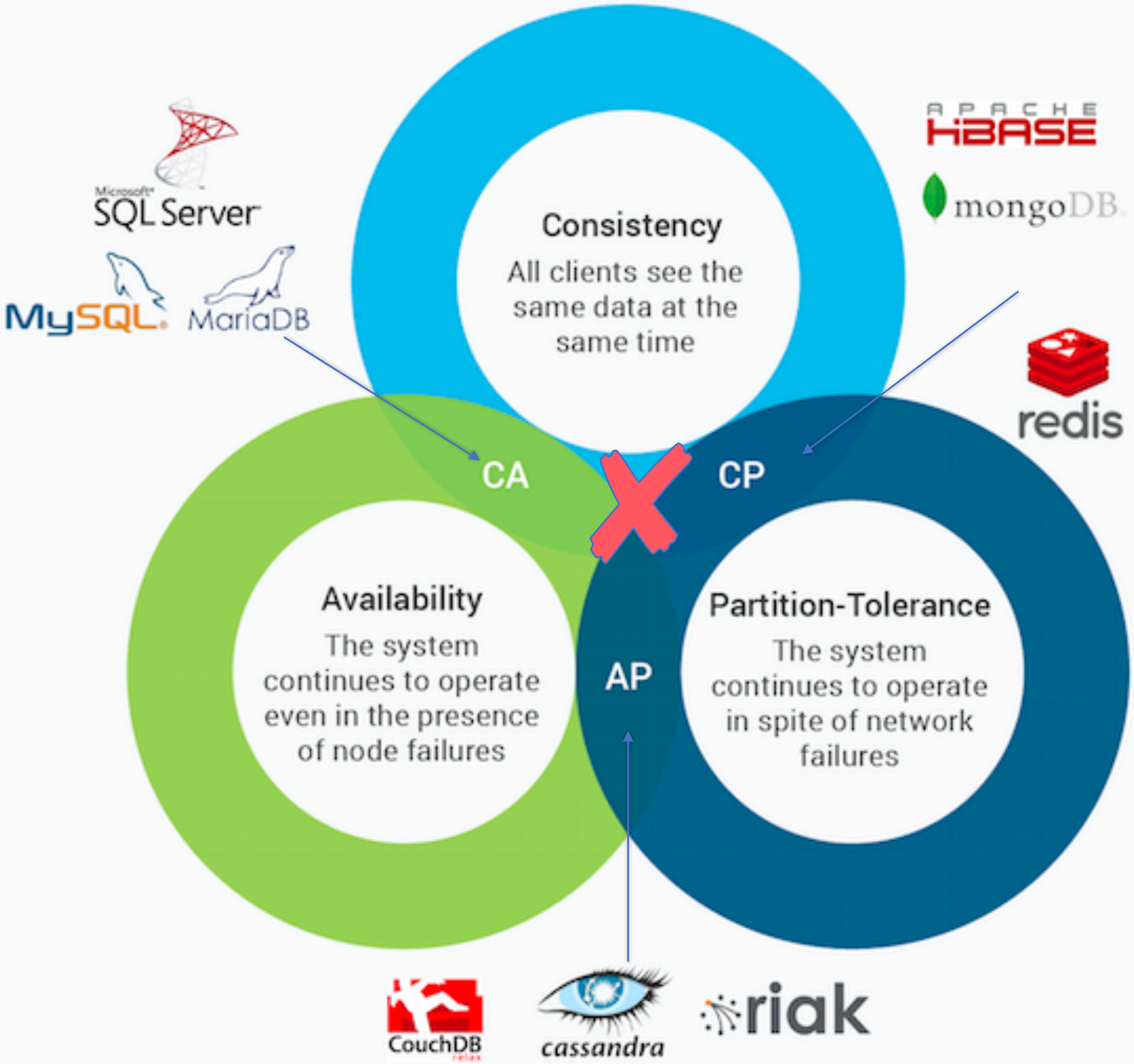
Guja Nemsadze

საკითხები

- What is the CAP theorem?
- What is the PACELC theorem?
- PACELC by DB
- ACID / BASE
- Amazon Aurora
- Amazon RDS
- Common data categories and use cases
- AWS database services
- AWS DMS and AWS SCT
- Hands on RDS
- Connect to RDS DB using https://dbeaver.io/download/

What is the CAP theorem?

The CAP theorem maintains that a distributed system can deliver only two of three desired characteristics: **C**onsistency, **A**vailability, and **P**artition tolerance.

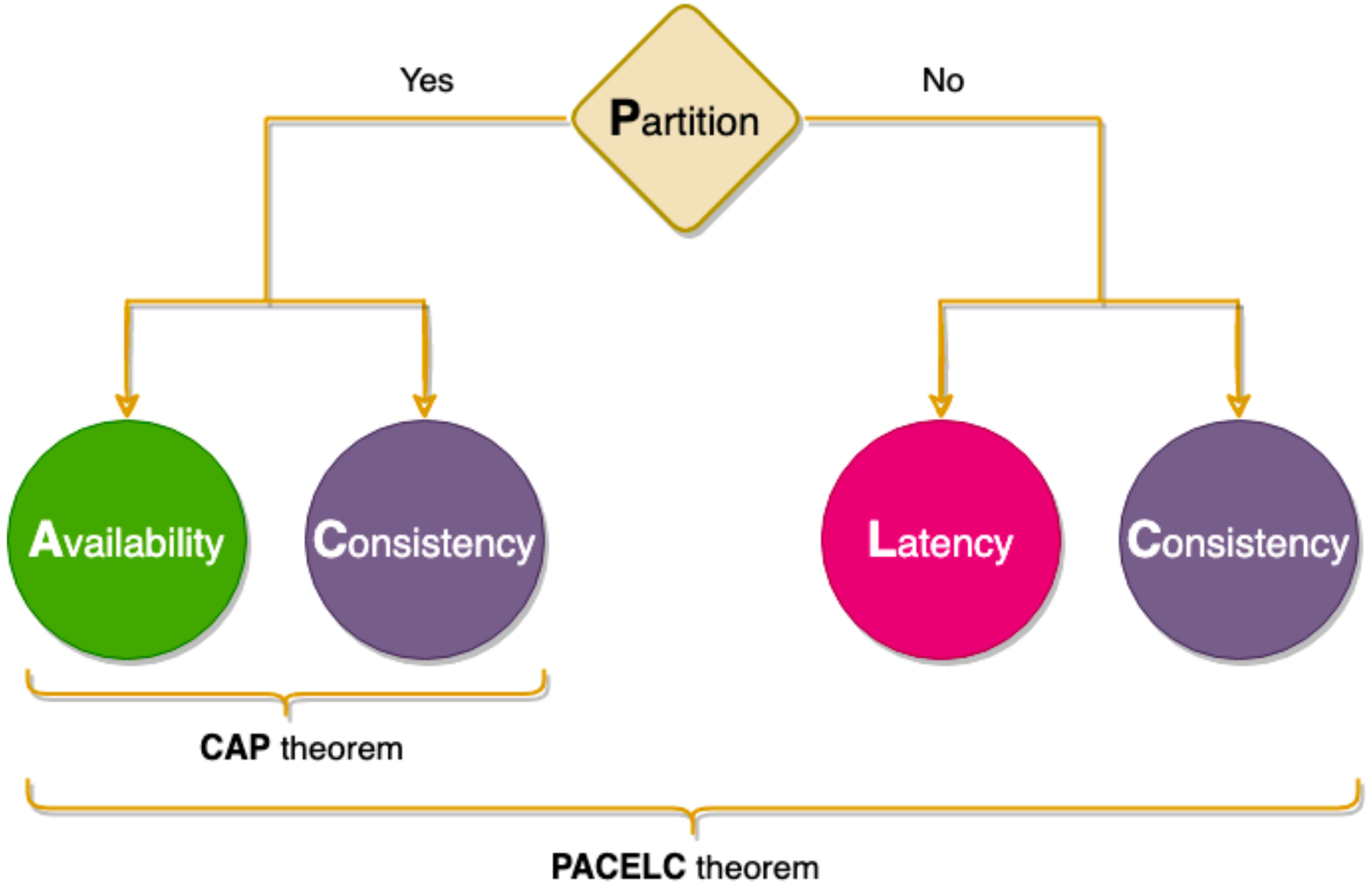

Microsoft SQL Server
MySQL
MariaDB
APACHE HBASE
mongoDB.
redis
Consistency
All clients see the same data at the same time
CA
CP
AP
Availability
The system continues to operate even in the presence of node failures
Partition-Tolerance
The system continues to operate in spite of network failures
CouchDB
cassandra
riak

What is the CAP theorem?

- !! ***CAP*** *theorem states that it is impossible for a distributed system to simultaneously provide all three of the following desirable properties* !!
- **Consistency** (**C**): All nodes see the same data at the same time. This means users can read or write from/to any node in the system and will receive the same data. **It is equivalent to having a single up-to-date copy of the data.**
- **Availability (A):** In simple terms, **availability refers to a system's ability to remain accessible even if one or more nodes in the system go down.**
- **Partition tolerance (P):** A partition-tolerant system continues to operate even if there are partitions in the system. **Such a system can sustain any network failure that does not result in the failure of the entire network. Data is sufficiently replicated across combinations of nodes and networks to keep the system up through intermittent outages.**

What is the PACELC theorem?

One place where the CAP theorem is silent is what happens when there is no network partition? What choices does a distributed system have when there is no partition?



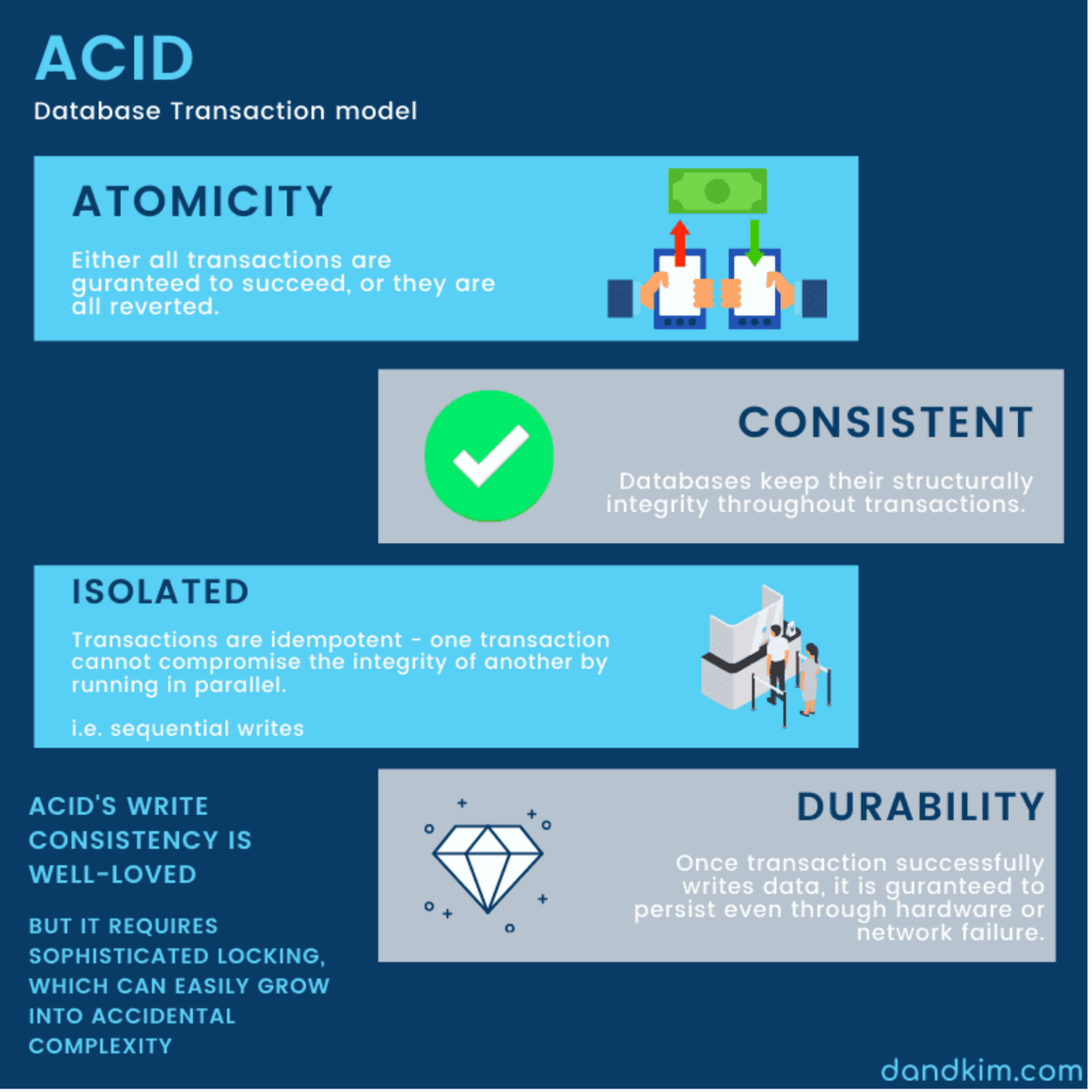
What is the PACELC theorem?

- if there is a partition ('P'), a distributed system can tradeoff between availability and consistency (i.e., 'A' and 'C');
- else ('E'), when the system is running normally in the absence of partitions, the system can tradeoff between latency ('L') and consistency ('C').

So, when there is a failure, CAP theorem prevails. But if not, we still have to consider the tradeoff between consistency and latency of a replicated system.

PACELC by DB

- **Dynamo and Cassandra** are **PA/EL** systems: They choose **availability** over **consistency** when a partition occurs; otherwise, they choose lower latency.
- **BigTable and HBase** are **PC/EC** systems: They will always choose **consistency**, giving up availability and lower latency.
- **MongoDB** can be considered **PA/EC** (default configuration): MongoDB works in a primary/secondaries configuration. In the default configuration, all writes and reads are performed on the primary. As all replication is done asynchronously (from primary to secondaries), when there is a network partition in which primary is lost or becomes isolated on the minority side, there is a chance of losing data that is unreplicated to secondaries, hence there is a loss of consistency during partitions. Therefore it can be concluded that in the case of a network partition, MongoDB chooses **availability**, but otherwise guarantees **consistency**. Alternately, when MongoDB is configured to write on majority replicas and read from the primary, it could be categorized as **PC/EC**.



Examples

Relational database systems are usually ***ACID***.

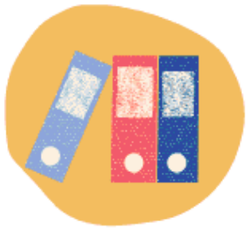
MySQL, PostgreSQL, SQLite, SQL Server, etc.

BASE

Database Transaction Model

Basically Available

Appears to work most of the time



Soft State

Stores may not be write-consistent. Replicas may not be mutually consistent with each other.

Eventual Consistency

Writes will respond with SUCCESS before the data is actually persisted to disk. This is one of the factors that contribute to the huge performance benefit.

The drawback? You might not see the data reflected in time.

Key Advantage

Some applications don't benefit from the strict consistent of the ACID model.

For example, Twitter users may not mind seeing their tweet take a short moment before seeing it posted.

Make sure it fits your use-case

A popular example of when NOT to use BASE: financial transactions.

Every single transaction is important.

Every single transaction needs to be properly reflected and kept consistent throughout the entire system.

Examples

NoSQL databases tend to be ***BASE***.

MongoDB, Cassandra, Redis, Amazon DynamoDB, Couchbase, etc.

Amazon Aurora

+

Commercial-grade performance and reliability ?

Amazon Aurora

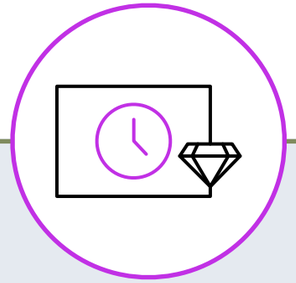
MySQL and PostgreSQL-compatible relational database built for the cloud

Performance and availability of commercial-grade databases at 1/10th the cost

Performance and scalability

5x throughput of standard MySQL and 3x of standard PostgreSQL; scale-out up to 15 read replicas

Availability and durability



Fault-tolerant, self-healing storage; six copies of data across three Availability Zones; continuous backup to Amazon S3

Highly secure

Network isolation, encryption at rest/transit

Fully managed

Managed by RDS: No hardware provisioning, software patching, setup, configuration, or backups

Amazon Aurora

Customer success story

"At the UN, we operate multiple websites with global reach that require mission-critical reliability and consistent performance.

We were able to achieve superb performance even with Amazon Aurora's smallest database engine.

Amazon Aurora's new user-friendly monitoring interface made it easy to diagnose and address issues.

Its performance, reliability and monitoring really shows Amazon Aurora is an enterprise-grade AWS database"

Mohamad Reza, Information Systems Officer - United Nations

Amazon Relational Database Service (RDS)

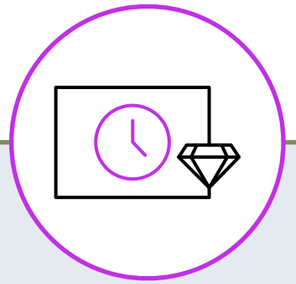
Managed relational database service with a choice of six popular database engines

Amazon Aurora · MySQL · PostgreSQL · MariaDB · Microsoft SQL Server · ORACLE

Easy to administer

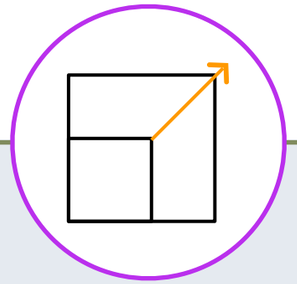
No need for infrastructure provisioning, installing, and maintaining DB software

Available and durable



Automatic Multi-AZ data replication; automated backup, snapshots, failover

Highly scalable



Scale database compute and storage with a few clicks with no app downtime

Fast and secure

SSD storage and guaranteed provisioned I/O; data encryption at rest and in transit

Remember!

Aurora’s unique architecture gives you more durability, scalability, resiliency, and performance when compared to RDS. Although there is a small increase in cost, it is recommend using Aurora for enterprise-level applications. If you are looking for a native high availability solution and/or read-intensive workload, then Aurora is a perfect match.

Common data categories and use cases

Relational

Referential integrity, ACID transactions, schema-on-write

Common Use Cases

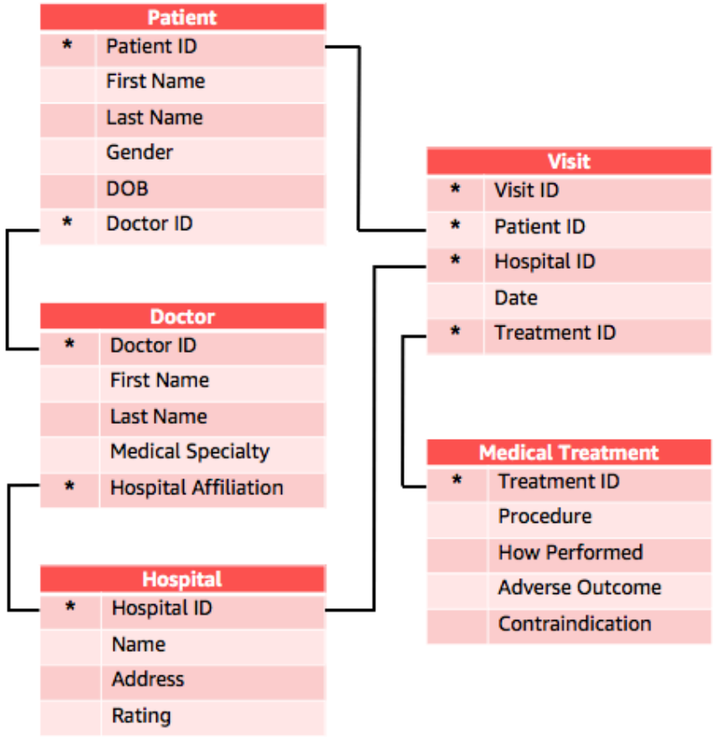
Lift and shift, ERP, CRM, finance

AWS Service(s)

Aurora, RDS

Relational data

- Divide data among tables
- Highly structured
- Relationships established via keys enforced by the system
- Data accuracy and consistency



Common data categories and use cases

Relational

Key-value

	Relational	Key-value
	Referential integrity, ACID transactions, schema-on-write	High throughput, low-latency reads and writes, endless scale
Common Use Cases	Lift and shift, ERP, CRM, finance	Real-time bidding, shopping cart, social, product catalog, customer preferences
AWS Service(s)	**Aurora, RDS**	**DynamoDB**

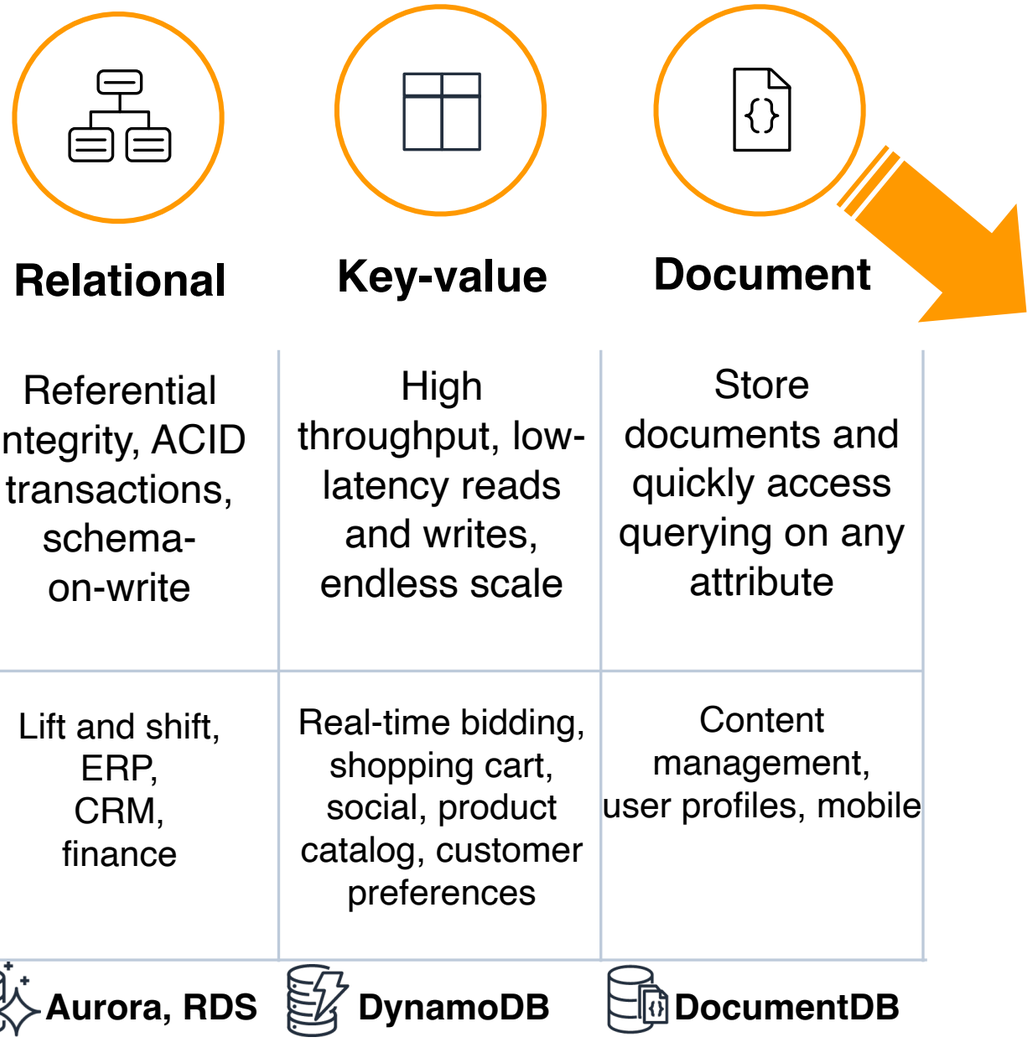
Key-value data

Key-value use case

```
// Status of Hammer57
GET {
  TableName:"Gamers",
  Key: {
    "GamerTag":"Hammer57",
    "Type":"Status" } }
```

Gamers				
Primary Key		**Attributes**		
Gamer Tag	**Type**			
Hammer57	Rank	Level	Points	Tier
		87	4050	Elite
	Status	Health	Progress	
		90	30	
	Weapon	Class	Damage	Range
		Taser	87%	50
FluffyDuffy	Rank	Level	Points	Tier
		5	1072	Trainee
	Status	Health	Progress	
		37	8	

Common data categories and use cases



	Relational	Key-value	Document
	Referential integrity, ACID transactions, schema-on-write	High throughput, low-latency reads and writes, endless scale	Store documents and quickly access querying on any attribute
Common Use Cases	Lift and shift, ERP, CRM, finance	Real-time bidding, shopping cart, social, product catalog, customer preferences	Content management, user profiles, mobile
AWS Service(s)	**Aurora, RDS**	**DynamoDB**	**DocumentDB**

Document databases

- Data is stored in JSON-like documents
- Documents map naturally to how humans model data
- Flexible schema and indexing
- Expressive query language built for documents (ad hoc queries and aggregations)

JSON documents are first-class objects of the database

```
{
  id: 1,
  name: "sue",
  age: 26,
  email: "sue@example.com",
  promotions: ["new user", "5%", "dog lover"],
  memberDate: 2018-2-22,
  shoppingCart: [
    {product:"abc", quantity:2, cost:19.99},
    {product:"edf", quantity:3, cost: 2.99}
  ]
}
```

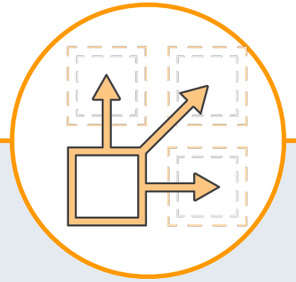
Amazon DocumentDB

Fast, scalable, and fully managed MongoDB-compatible database service

Fast

Millions of requests per second with millisecond latency; twice the throughput of MongoDB

Scalable



Separation of compute and storage enables both layers to scale independently; scale out to 15 read replicas in minutes

Fully managed

Managed by AWS:
no hardware provisioning; auto patching, quick setup, secure, and automatic backups

MongoDB compatible

Compatible with MongoDB 3.6; use the same SDKs, tools, and applications with Amazon DocumentDB

https://www.mongodb.com/compare/documentdb-vs-mongodb

Common data categories and use cases

	Relational	Key-value	Document	In-memory
	Referential integrity, ACID transactions, schema-on-write	High throughput, low-latency reads and writes, endless scale	Store documents and quickly access querying on any attribute	Query by key with microsecond latency
Common Use Cases	Lift and shift, ERP, CRM, finance	Real-time bidding, shopping cart, social, product catalog, customer preferences	Content management, personalization, mobile	Leaderboards, real-time analytics, caching
AWS Service(s)	**Aurora, RDS**	**DynamoDB**	**DocumentDB**	**ElastiCache**

Amazon ElastiCache

Redis and Memcached compatible, in-memory data store and cache

Redis & Memcached compatible

Fully compatible with open source Redis and Memcached

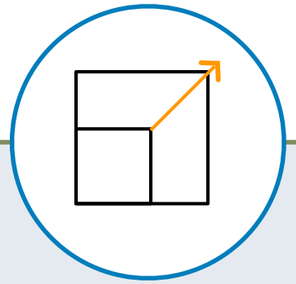
Extreme performance

In-memory data store and cache for microsecond response times

Secure and reliable

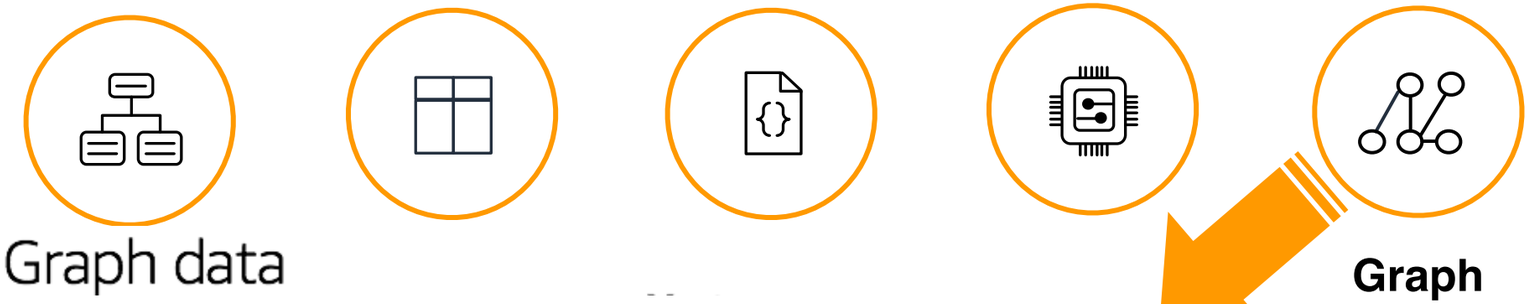
Network isolation, encryption at rest/transit, HIPAA, PCI, FedRAMP, multi AZ, and automatic failover

Easily scalable



Scale writes and reads with sharding and replicas

Common data categories and use cases



Graph data

Graph

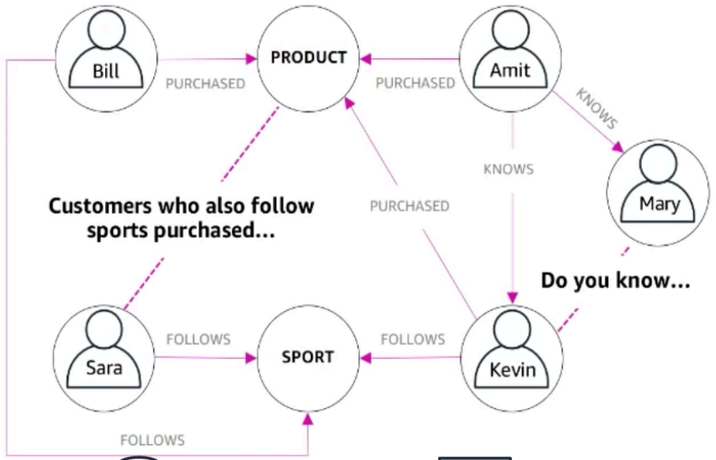
Graph use case

// Product recommendation to a user

```
gremlin> g.V().has('name','sara').as('customer').out('follows').in('follows').out('purchased')
where(neq('customer')).dedup().by('name').properties('name')
```

// Identify a friend in common and make a recommendation

```
gremlin> g.V().has('name','mary').as('start').
both('knows').both('knows').
where(neq('start')).
dedup().by('name').properties('name')
```



Quickly and easily create and navigate relationships between data

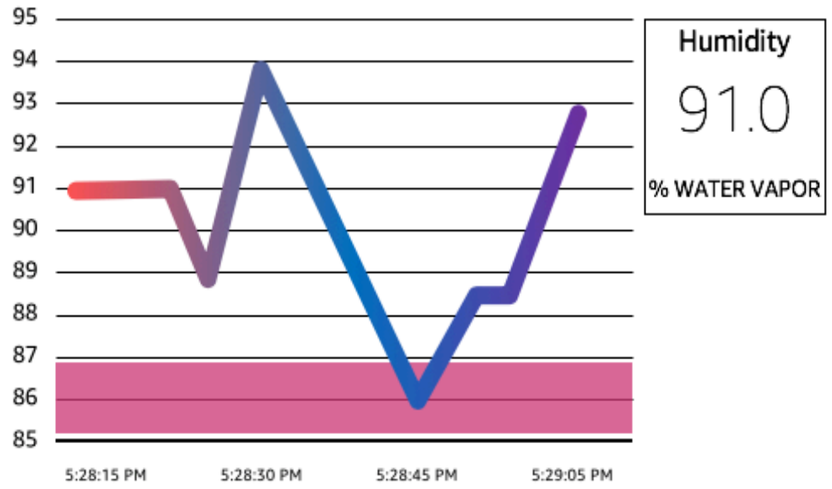
Fraud detection, social networking, recommendation engine

Comm
Use Cas

AWS Service(s)

Aurora, RDS **DynamoDB** **DocumentDB** **ElastiCache** **Neptune**

Common data categories and use cases

Relational	Key-value	Document	In-memory	Graph	Time-series
Referenti integrity, AC transactior schema- on-write				① Application events ② IoT Sensor Readings ③ Vehicle Telematics	Collect, store, and process data sequenced by time
Common Use Cases Lift and shift, CRM, finan					IoT applications, event tracking
AWS Service(s) Aurora, RDS	DynamoDB	DocumentDB	ElastiCache	Neptune	Timestream

Amazon Timestream (sign up for the preview)

Fast, scalable, fully managed time-series database

1,000x faster and 1/10th the cost of relational databases

Collect data at the rate of millions of inserts per second (10M/second)

Trillions of daily events

Adaptive query processing engine maintains steady, predictable performance

Time-series analytics

Built-in functions for interpolation, smoothing, and approximation

Serverless

Automated setup, configuration, server provisioning, software patching

Common data categories and use cases

	Relational	Key-value	Document	In-memory	Graph	Time-series	Ledger
	Referential integrity, ACID transactions, schema-on-write	High throughput, low-latency reads and writes, endless scale	Store documents and quickly access querying on any attribute	Query by key with microsecond latency	Quickly and easily create and navigate relationships between data	Collect, store, and process data sequenced by time	Complete, immutable, and verifiable history of all changes to application data
Common Use Cases	Lift and shift, ERP, CRM, finance	Real-time bidding, shopping cart, social, product catalog, customer preferences	Content management, personalization, mobile	Leaderboards, real-time analytics, caching	Fraud detection, social networking, recommendation engine	IoT applications, event tracking	Systems of record, supply chain, health care, registrations, financial
AWS Service(s)	**Aurora, RDS**	**DynamoDB**	**DocumentDB**	**ElastiCache**	**Neptune**	**Timestream**	**QLDB**

BTU

AWS database services

Purpose-built databases, the right tool for the right job

Relational

- Aurora
  - MySQL
  - PostgreSQL
- RDS
  - MySQL
  - PostgreSQL
  - MariaDB
  - ORACLE
  - Microsoft SQL Server
- Redshift

Non-relational

- DynamoDB — Key value
- ElastiCache — In-memory
- Neptune — Graph
- DocumentDB — Document
- Timestream — Time series
- QLDB — Ledger

Database Migration Service

AWS DMS and AWS SCT

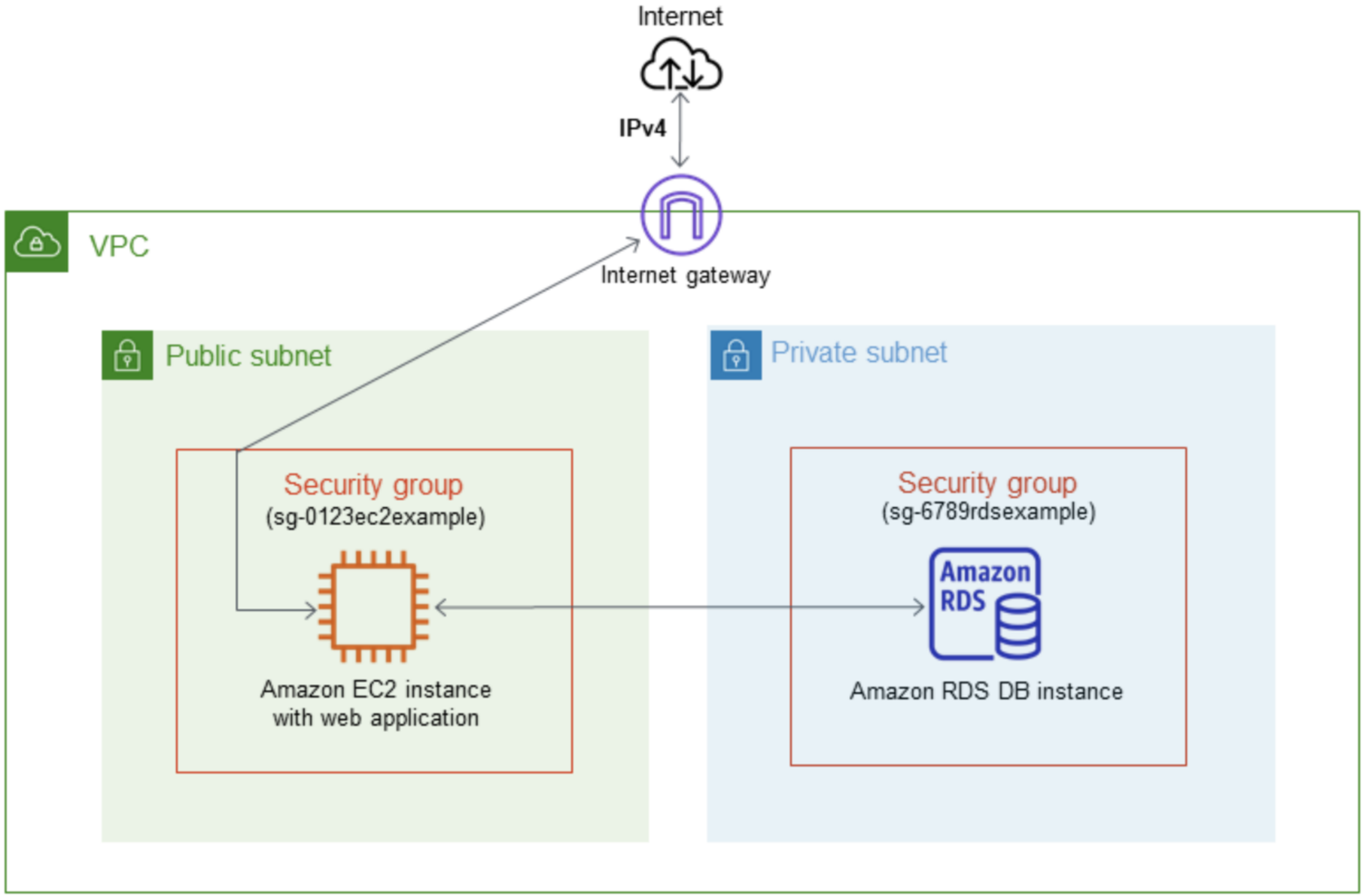
Our goal: Allow customers the freedom to choose the best data platform for their needs #DBFreedom

AWS SCT converts your commercial database and data warehouse schemas to open-source engines or AWS-native services, such as Amazon Aurora and Redshift

AWS DMS easily and securely migrates and/or replicate your databases *and* data warehouses to AWS

Hands on RDS

- Connect to RDS DB using https://dbeaver.io/download/
- Seed Mysql RDS with movie data https://replit.com/@JexPY/WatchfulAliceblueEmulation
- *At home* CRUD DB script https://replit.com/@JexPY/MatureDistantSubversion#main.py


Internet
IPv4
Internet gateway
VPC
Public subnet
Private subnet
Security group
(sg-0123ec2example)
Amazon EC2 instance
with web application
Security group
(sg-6789rdsexample)
Amazon
RDS
Amazon RDS DB instance

ლიტერატურა

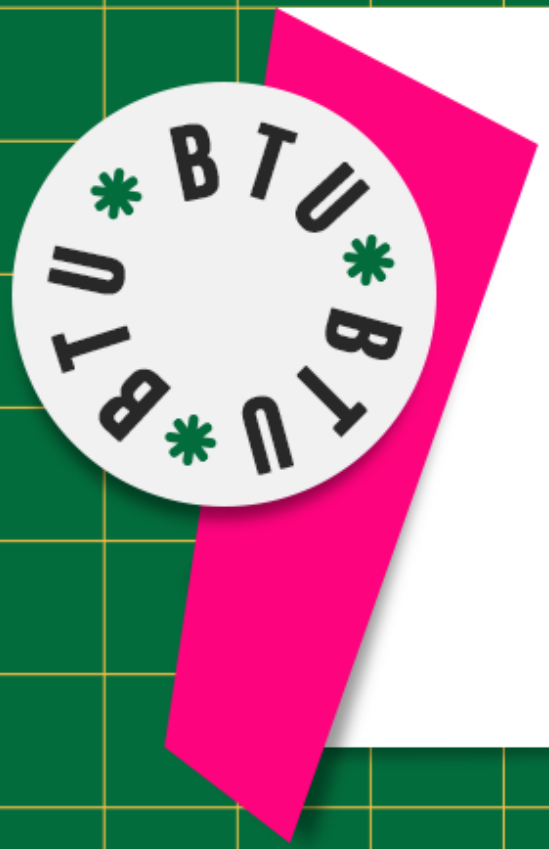
1. AWS Scripted: How to Automate the Deployment of Secure and Resilient Websites with Amazon Web Services VPC, ELB, EC2, RDS, IAM, SES and SNS - კრისტიან ჩერი, 2014წ. , 270 გვერდი
2. Amazon Web Services in Action - ანდრეას ვიტიგი, მიქაელ ვიტიგი, 2016წ., 424 გვ,
3. https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/ec2.html

მადლობა ყურადღებისთვის!

კონტაქტი

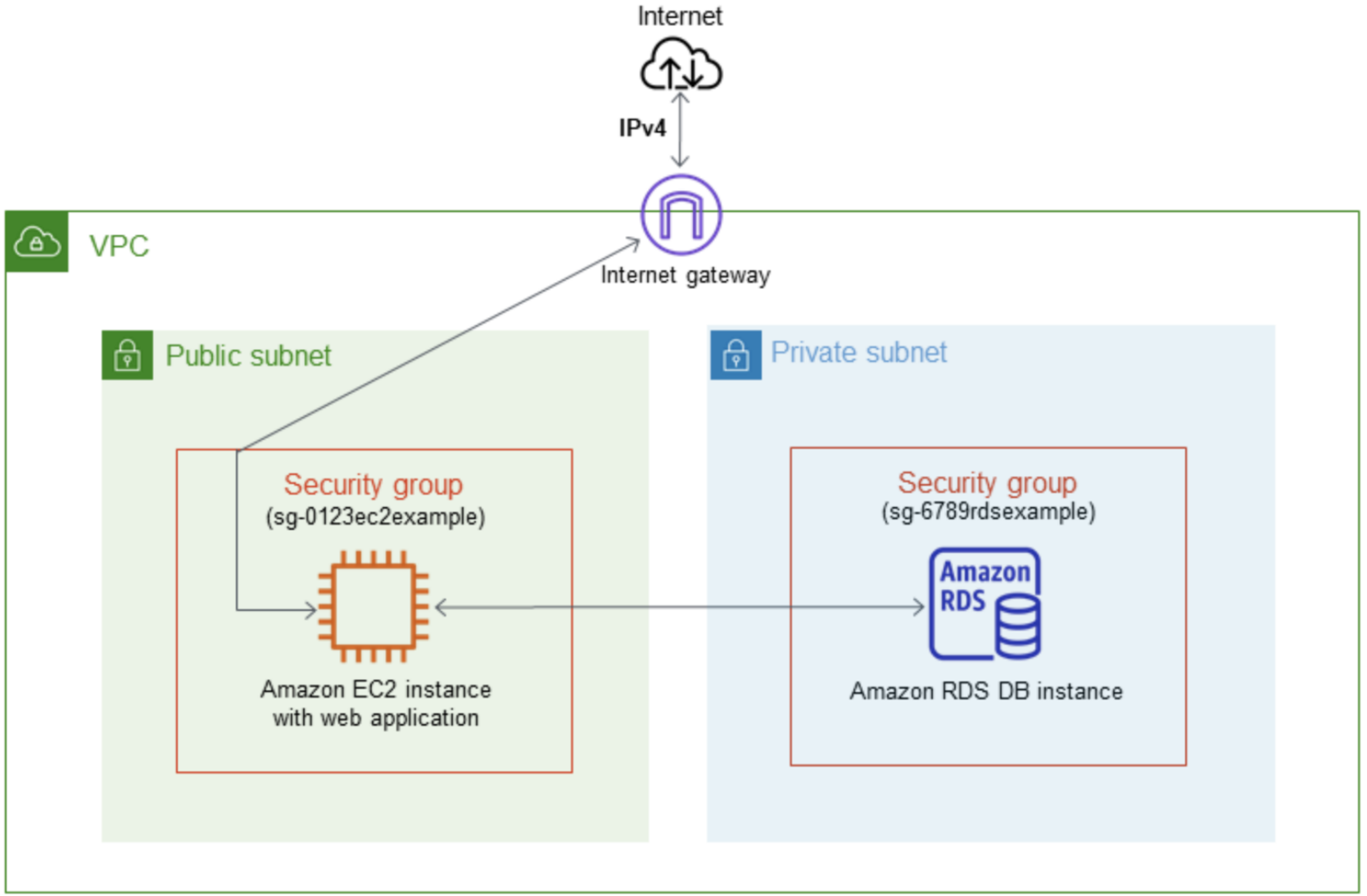
თბილისი 0162, საქართველო
ი.ჭავჭავაძის გამზირი 82
INFO@BTU.EDU.GE
032 2 195 015
WWW.BTU.EDU.GE

გამოგვყევი:



ღრუბლოვანი სისტემების (AWS) ავტომატიზაცია Python-ის საშუალებით

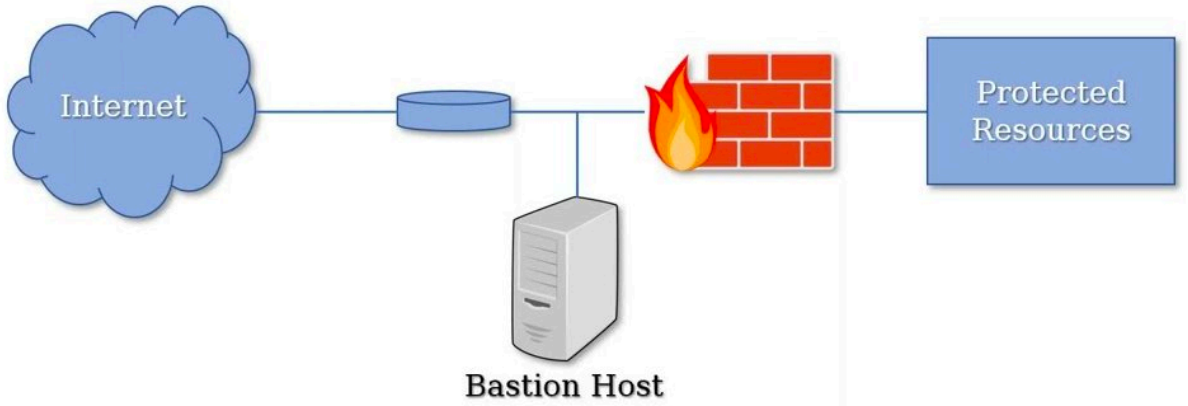
Guja Nemsadze


Internet
IPv4
Internet gateway
VPC
Public subnet
Private subnet
Security group
(sg-0123ec2example)
Amazon EC2 instance
with web application
Security group
(sg-6789rdsexample)
Amazon
RDS
Amazon RDS DB instance

საკითხები

- What is a Bastion Host?
- What Are the Types of Bastion Hosts?
- Hands On Bastion Host.
- Run queries on RDS Postgres using DBeaver.
-

What is a Bastion Host?



A **bastion host** is a server used to manage access to an internal or private network from an external network - sometimes called a **jump box** or **jump server**. Because bastion hosts often sit on the Internet, they *typically run a minimum amount of services in order to reduce their attack surface*. They are also commonly used to proxy and log communications, such as SSH sessions.

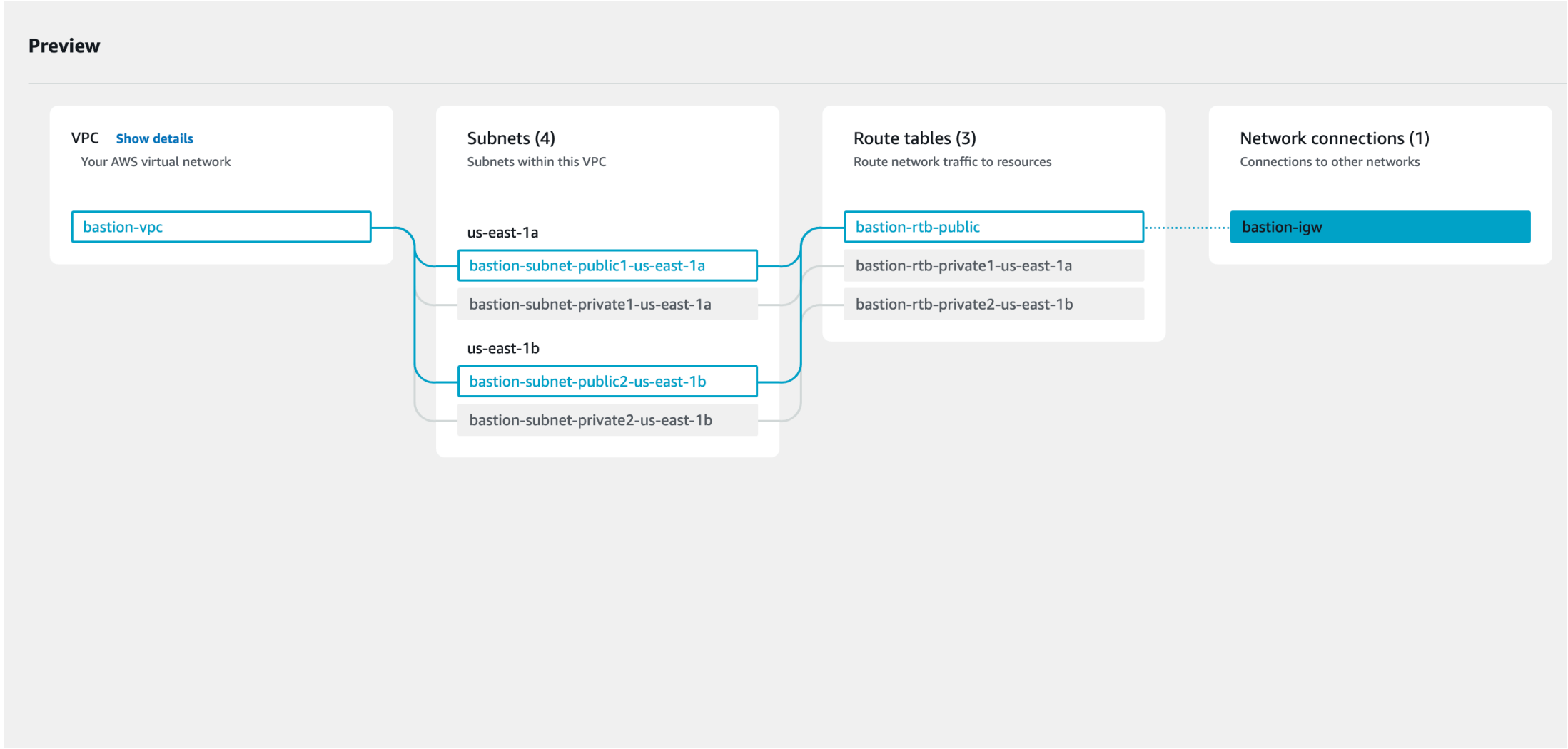
What Are the Types of Bastion Hosts?

- Domain Name System (DNS) server
- Email server
- File Transfer Protocol (FTP) server
- Honeypot
- Proxy server
- Virtual private network (VPN) server
- Web server

Hands On Bastion Host.

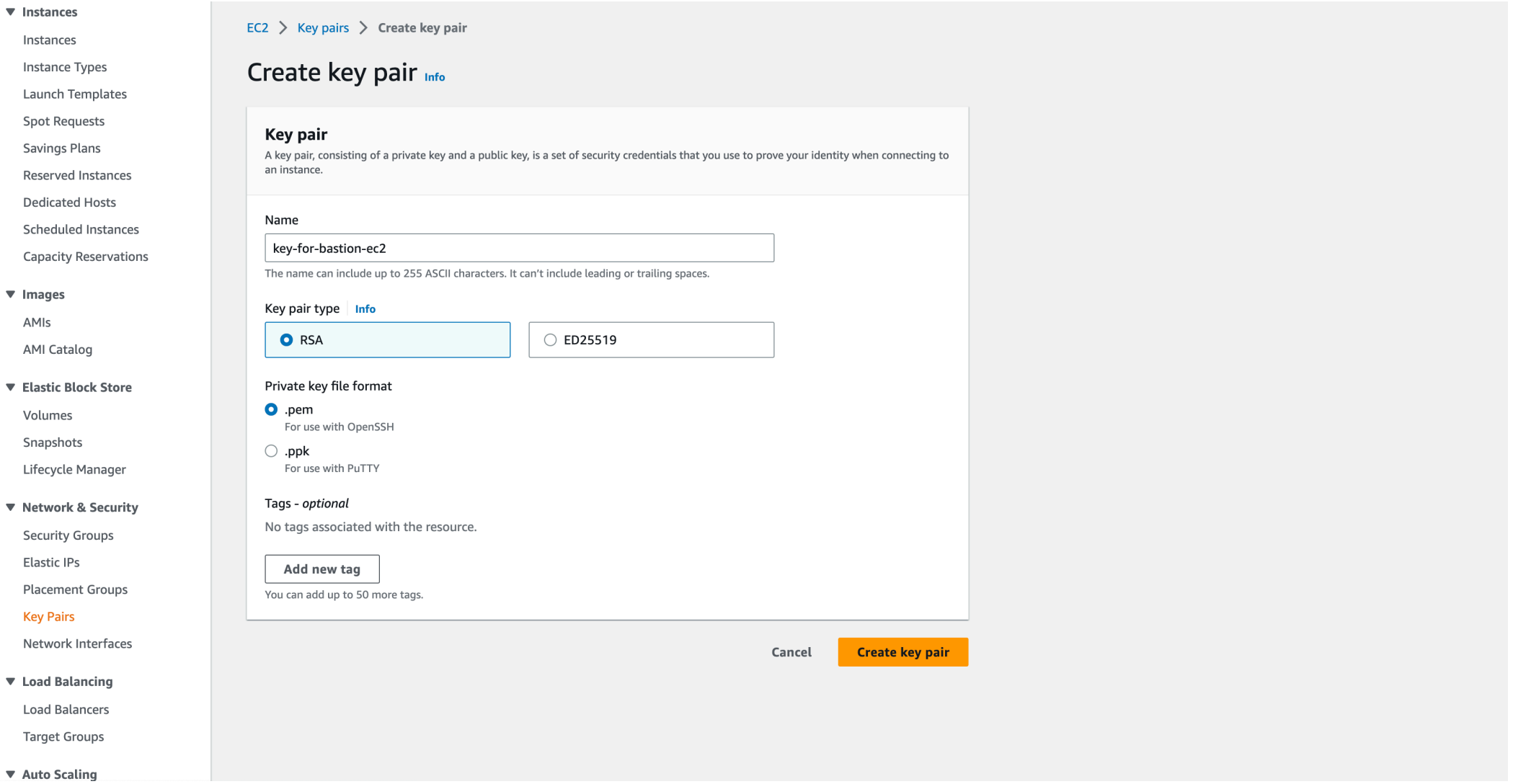
Create VPC

Hands On Bastion Host.



Create Key pair

Hands On Bastion Host.



Create Launch EC2 instance

Hands On Bastion Host.

Name and tags Info

Name

my-bastion-ec2

Add additional tags

Application and OS Images (Amazon Machine Image) Info

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Search our full catalog including 1000s of application and OS images

Recents | Quick Start

Amazon Linux | macOS | Ubuntu | Windows | Red Hat | S

Browse more AMIs

Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Amazon Linux 2023 AMI — Free tier eligible

ami-0715c1897453cabd1 (64-bit (x86)) / ami-041c36ce1b70dfc41 (64-bit (Arm))

Virtualization: hvm ENA enabled: true Root device type: ebs

Description

Amazon Linux 2023 AMI 2023.0.20230517.1 x86_64 HVM kernel-6.1

Architecture: 64-bit (x86)

AMI ID: ami-0715c1897453cabd1 Verified provider

Instance type Info

Instance type

t2.micro — Free tier eligible

Family: t2 1 vCPU 1 GiB Memory Current generation: true

On-Demand Windows pricing: 0.0162 USD per Hour

On-Demand SUSE pricing: 0.0116 USD per Hour

On-Demand RHEL pricing: 0.0716 USD per Hour

On-Demand Linux pricing: 0.0116 USD per Hour

All generations

Compare instance types

Key pair (login) Info

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - *required*

key-for-bastion-ec2

Create new key pair

Network settings Info

VPC - *required* Info

vpc-0a910bd48f53812d1 (bastion-vpc)

10.0.0.0/16

Subnet Info

subnet-07c5421ec992c1a0c bastion-subnet-public1-us-east-1a

VPC: vpc-0a910bd48f53812d1 Owner: 319231157779

Availability Zone: us-east-1a IP addresses available: 4091 CIDR: 10.0.0.0/20)

Create new subnet

Auto-assign public IP Info

Enable

Firewall (security groups) Info

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

Create Launch EC2 instance

Hands On Bastion Host.

Firewall (security groups) Info

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

(•) Create security group ( ) Select existing security group

Security group name - *required*

launch-wizard-2

This security group will be added to all network interfaces. The name can't be edited after the security group is created. Max length is 255 characters. Valid characters: a-z, A-Z, 0-9, spaces, and ._-:/()#,@[]+=&;{}!$*

Description - *required* Info

launch-wizard-2 created 2023-06-01T17:33:51.244Z

Inbound security groups rules

▼ Security group rule 1 (TCP, 22, Multiple sources) — Remove

Type Info	Protocol Info	Port range Info
ssh	TCP	22

Source type Info	Source Info	Description - *optional* Info
Anywhere	*Add CIDR, prefix list or security*	*e.g. SSH for admin desktop*

0.0.0.0/0 ✕ ::/0 ✕

⚠ Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only. ✕

Add security group rule

▶ Advanced network configuration

▼ Summary

Number of instances Info

1

Software Image (AMI)

Amazon Linux 2023 AMI 2023.0.2...read more

ami-0715c1897453cabd1

Virtual server type (instance type)

t2.micro

Firewall (security group)

New security group

Storage (volumes)

1 volume(s) - 8 GiB

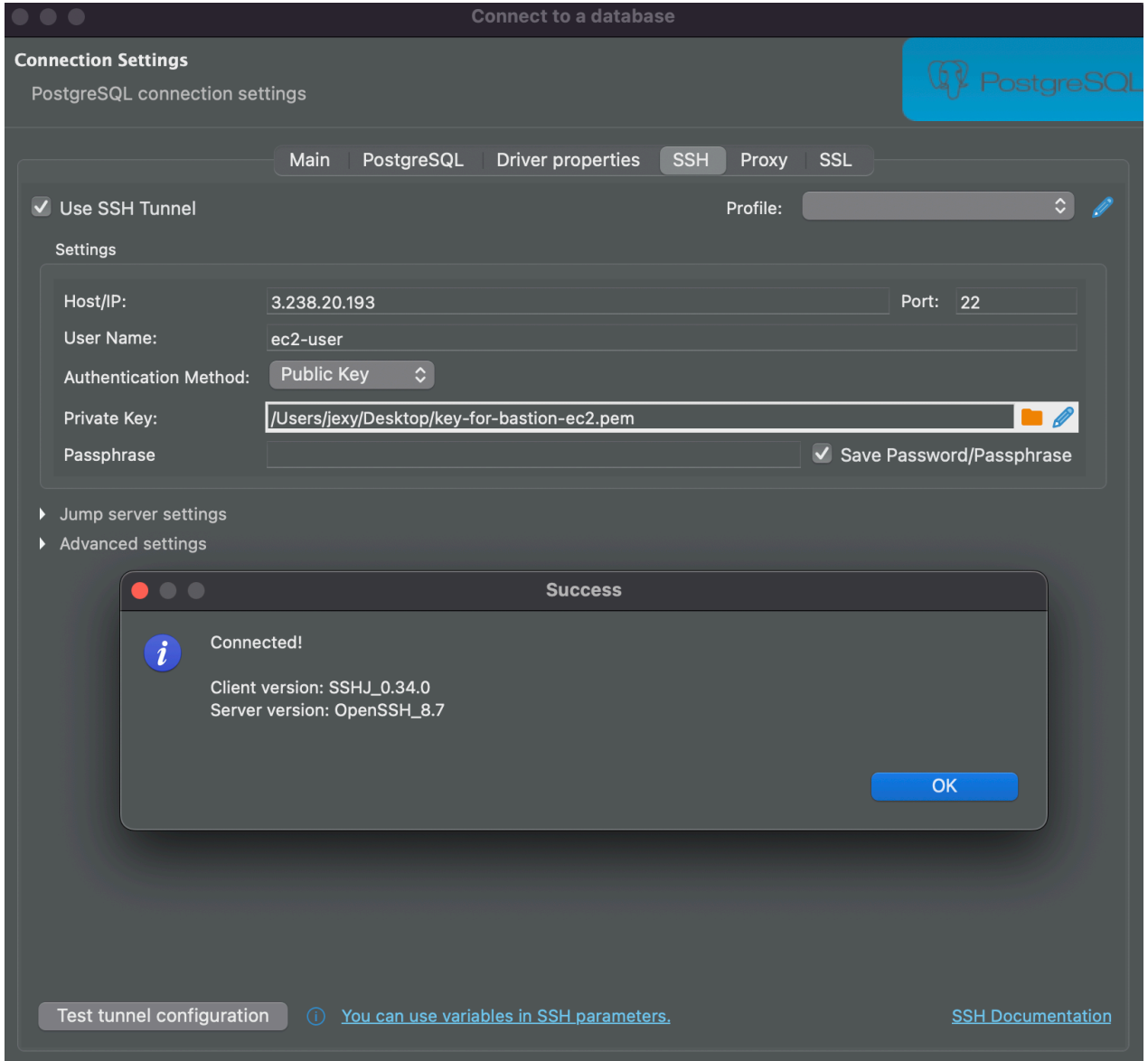
Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 30 GiB of EBS storage, 2 million IOs, 1 GB of snapshots, and 100 GB of bandwidth to the internet. ✕

Cancel **Launch instance**

Review commands

Test SSH Connection to EC2 instance Using **DBeaver**

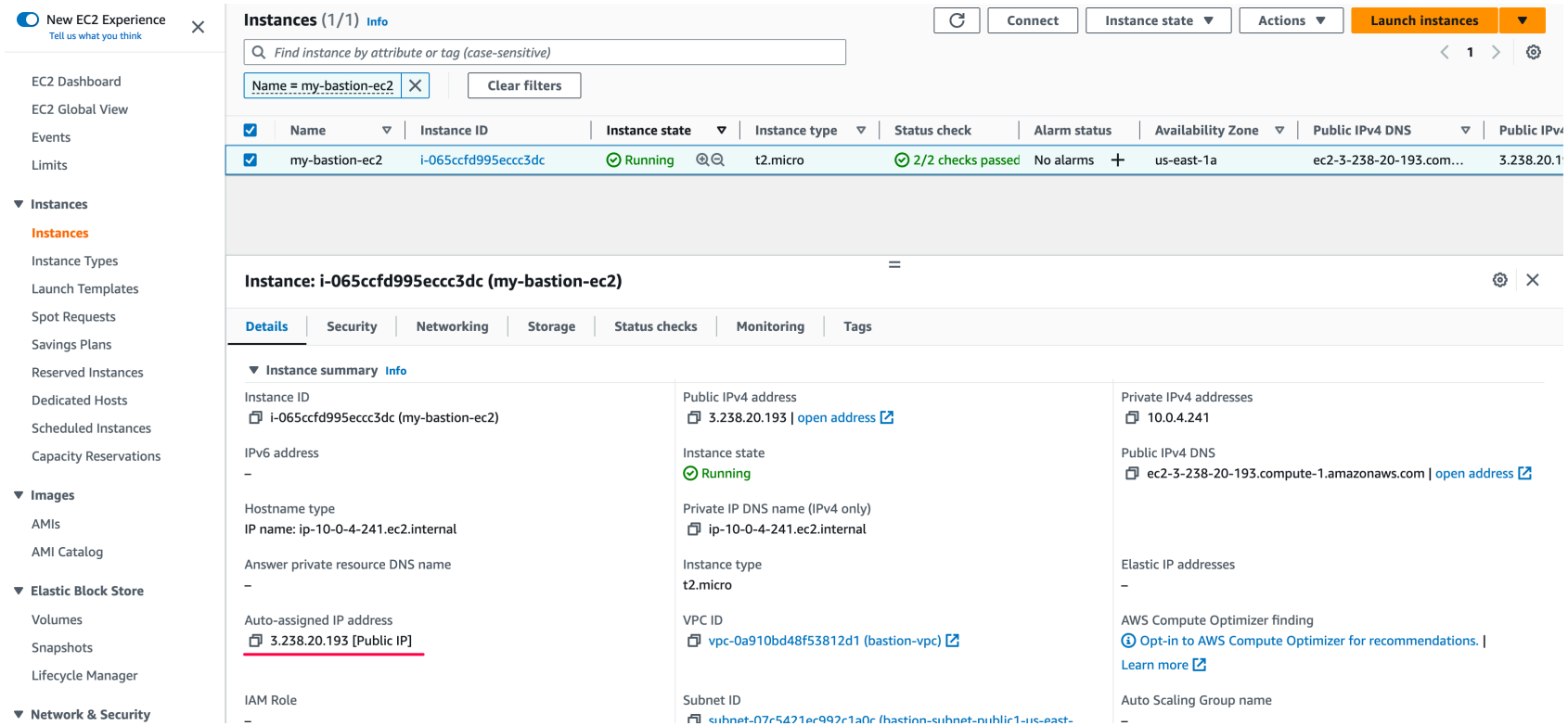
Hands On Bastion Host.



Test SSH Connection to EC2 instance Variables

Hands On Bastion Host.

Host/IP:



User Name:

https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/managing-users.html

Create DB Subnet Group

Hands On Bastion Host.

Subnet group details

Name

You won't be able to modify the name after your subnet group has been created.

private-sg-rds

Must contain from 1 to 255 characters. Alphanumeric characters, spaces, hyphens, underscores, and p

Description

This group will allow only private access.

VPC

Choose a VPC identifier that corresponds to the subnets you want to use for your DB subnet group. Yc
different VPC identifier after your subnet group has been created.

bastion-vpc (vpc-0a910bd48f53812d1)

Add subnets

Availability Zones

Choose the Availability Zones that include the subnets you want to add.

Choose an availability zone

us-east-1a ✕ us-east-1b ✕

Subnets

Choose the subnets that you want to add. The list includes the subnets in the selected Availability Zones.

Select subnets

- ☐ **us-east-1a**
  - ☐ subnet-07c5421ec992c1a0c (10.0.0.0/20)
  - ☐ subnet-062f59da2955ea958 (10.0.128.0/20)
- ☐ **us-east-1b**
  - ☐ subnet-0dc195bab7d73957c (10.0.16.0/20)
  - ☐ subnet-0bbfddb1a108b37a5 (10.0.144.0/20)

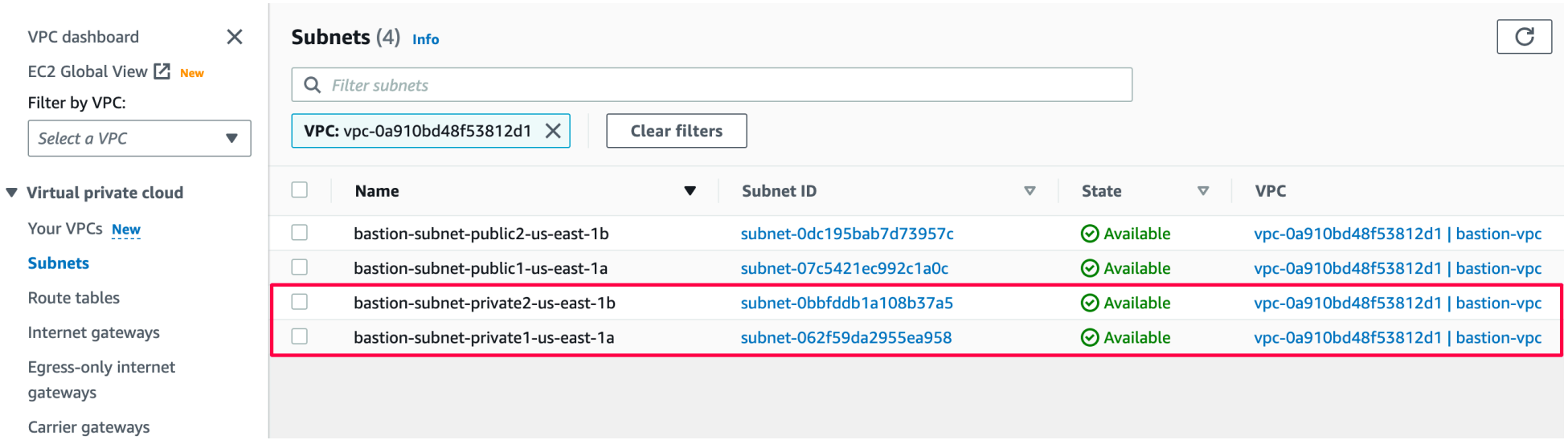
Zones.

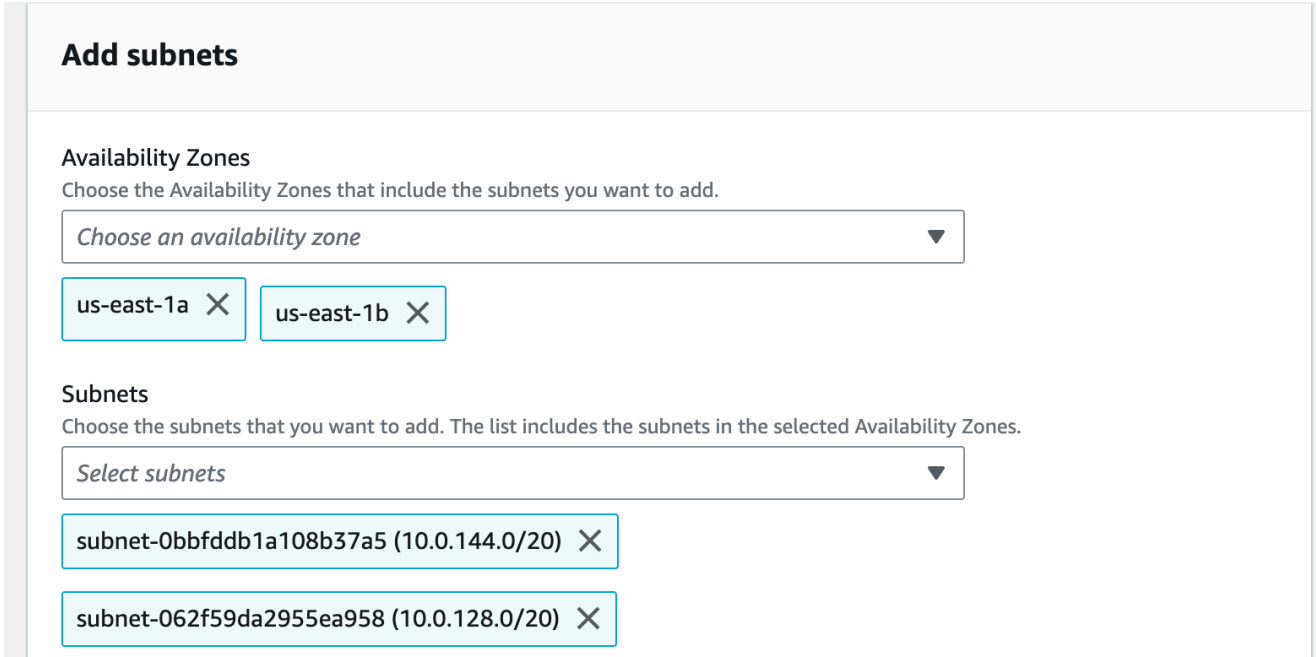
DR block

No subnets added to this group

Create DB Subnet Group

Hands On Bastion Host.





Create RDS/Postgres/Mysql/Aurora

Hands On Bastion Host.

Engine options

Engine type Info

- ○ Aurora (MySQL Compatible)
- ○ Aurora (PostgreSQL Compatible)
- ○ MariaDB
- ◉ PostgreSQL
- ○ Microsoft SQL Server

Connectivity Info

Compute resource

Choose whether to set up a connection to a compute resource for this database. Setting up a connection will automatically change connectivity settings so that the compute resource can connect to this database.

- ◉ Don't connect to an EC2 compute resource
  Don't set up a connection to a compute resource for this database. You can manually set up a connection to a compute resource later.
- ○ Connect to an EC2 compute resource
  Set up a connection to an EC2 compute resource for this database.

Virtual private cloud (VPC) Info

Choose the VPC. The VPC defines the virtual networking environment for this DB instance.

bastion-vpc (vpc-0a910bd48f53812d1)
4 Subnets, 2 Availability Zones

Only VPCs with a corresponding DB subnet group are listed.

After a database is created, you can't change its VPC.

DB subnet group Info

Choose the DB subnet group. The DB subnet group defines which subnets and IP ranges the DB instance can use in the VPC that you selected.

private-sg-rds
2 Subnets, 2 Availability Zones

Public access Info

- ○ Yes
  RDS assigns a public IP address to the database. Amazon EC2 instances and other resources outside of the VPC can connect to your database. Resources inside the VPC can also connect to the database. Choose one or more VPC security groups that specify which resources can connect to the database.
- ◉ No
  RDS doesn't assign a public IP address to the database. Only Amazon EC2 instances and other resources inside the VPC can connect to your database. Choose one or more VPC security groups that specify which resources can connect to the database.

Create RDS/Postgres/Mysql/Aurora

Hands On Bastion Host.

VPC security group (firewall) Info

Choose one or more VPC security groups to allow access to your database. Make sure that the security group rules allow the appropriate incoming traffic.

- ○ Choose existing
  Choose existing VPC security groups
- ◉ Create new
  Create new VPC security group

New VPC security group name

private-vpc-rds

Availability Zone Info

No preference

RDS Proxy

RDS Proxy is a fully managed, highly available database proxy that improves application scalability, resiliency, and security.

☐ Create an RDS Proxy Info

RDS automatically creates an IAM role and a Secrets Manager secret for the proxy. RDS Proxy has additional costs. For more information, see Amazon RDS Proxy pricing.

Certificate authority - *optional* Info

Using a server certificate provides an extra layer of security by validating that the connection is being made to an Amazon database. It does so by checking the server certificate that is automatically installed on all databases that you provision.

rds-ca-2019 (default)

If you don't select a certificate authority, RDS chooses one for you.

▼ Additional configuration

Database port Info

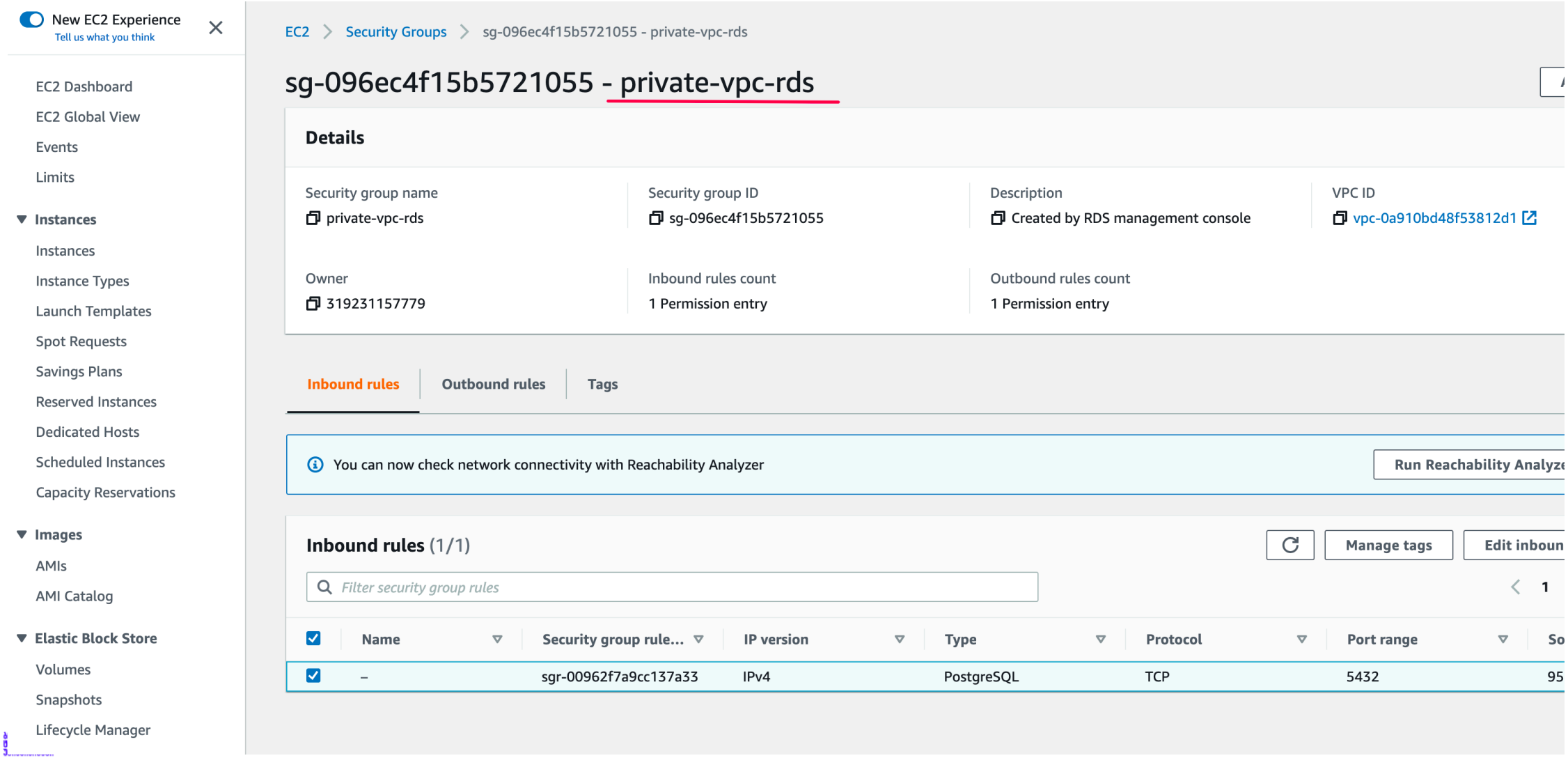
TCP/IP port that the database will use for application connections.

5432

Cancel | Create database

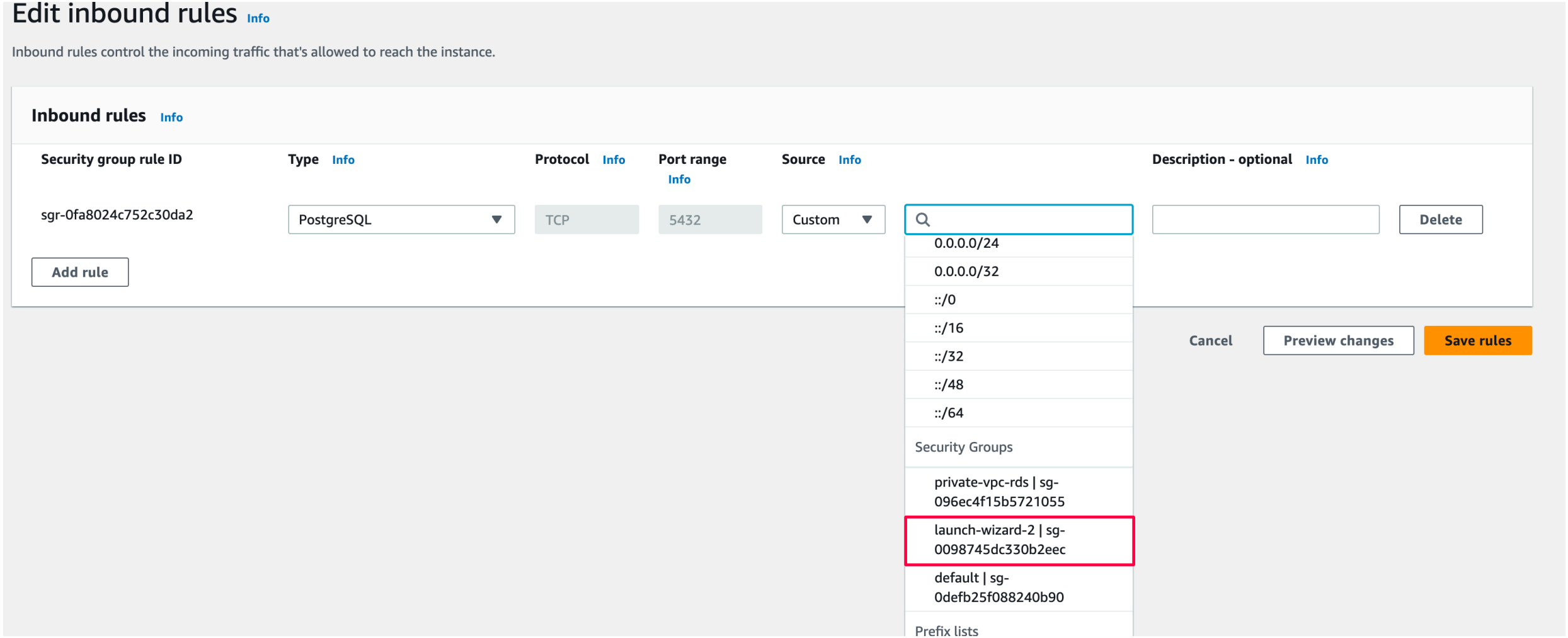
Edit RDS Security Group

Hands On Bastion Host.



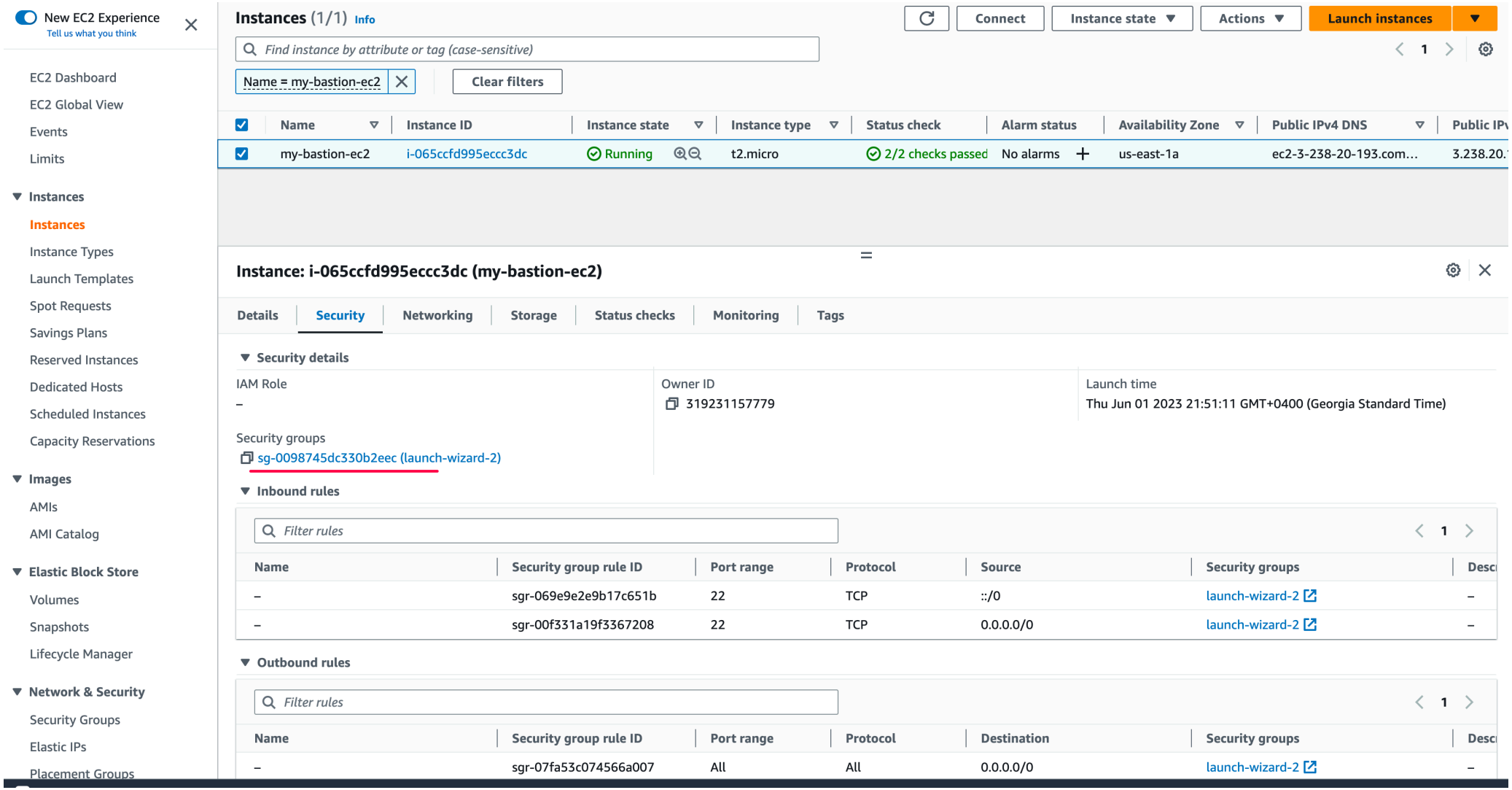
Edit RDS Security Group Inbound Rule

Hands On Bastion Host.



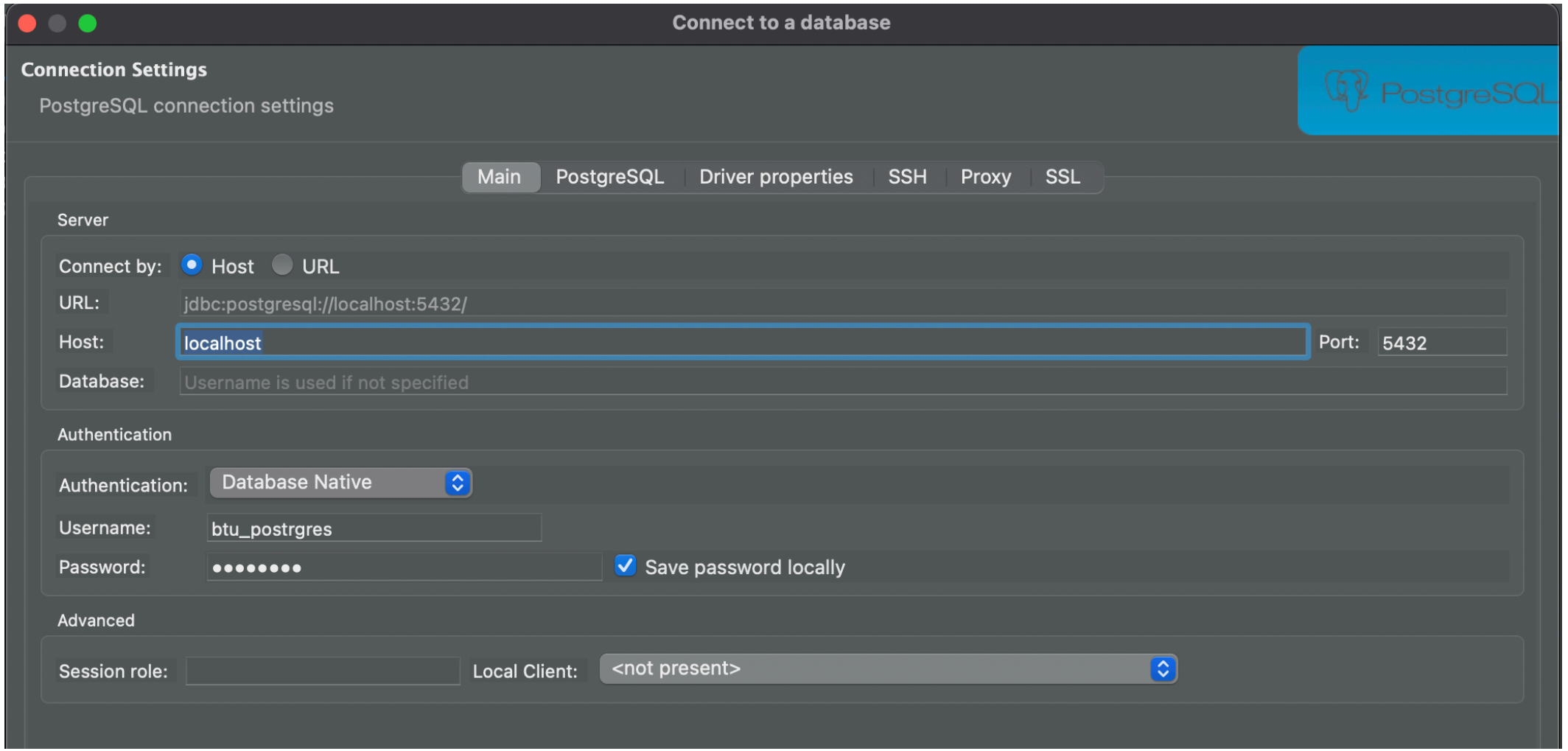
Edit RDS Security Group Inbound Rule

Hands On Bastion Host.



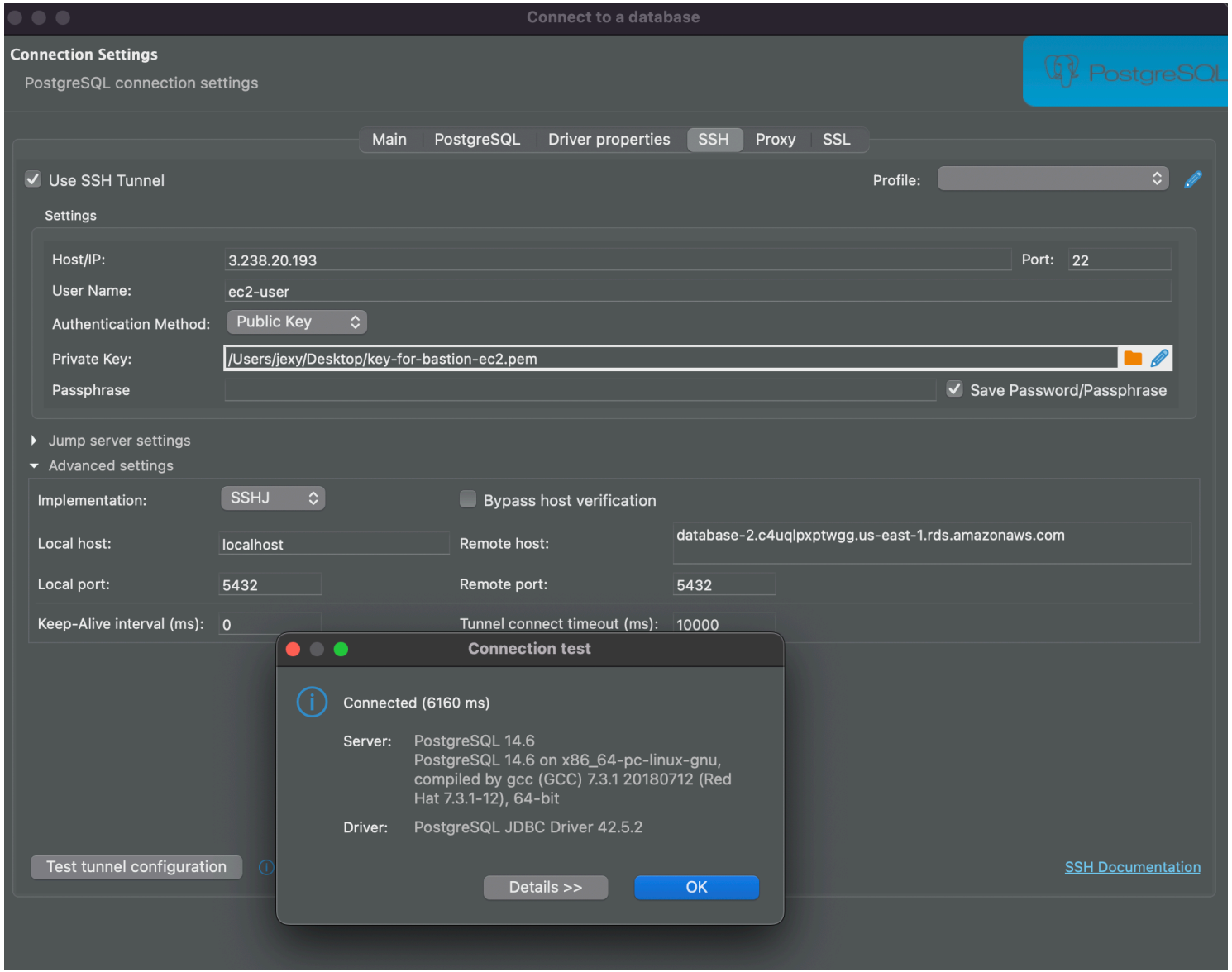
Connect to RDS Postgres
Using **DBeaver**

Hands On Bastion Host.



Connect to RDS Postgres Using **DBeaver**

Hands On Bastion Host.



Connect to RDS Postgres

Forming an SSH Tunnel

Hands On Bastion Host.

Example For Mysql:

```
ssh -i bastion-key.pem -f -N -L 3306:database-1.c4uqlpxptwgg.us-east-1.rds.amazonaws.com:3306 ec2-user@35.170.56.117 -v
```

Example For Postgre:

```
ssh -i key-for-bastion-ec2.pem -f -N -L 5432:database-2.c4uqlpxptwgg.us-east-1.rds.amazonaws.com:5432 ec2-user@3.238.20.193 -v
```

Run queries on RDS Postgres
Using DBeaver

- Simulate hospital data for PostgreSQL : https://pastebin.com/RfrGqKvz
- Reason of visits in Hospital : https://pastebin.com/cvB77KNK
- The average age of people who visited the hospital for each reason : https://pastebin.com/sJuVNiWP

Automated with Boto3

https://replit.com/@JexPY/DigitalDishonestMoto#main.py

ლიტერატურა

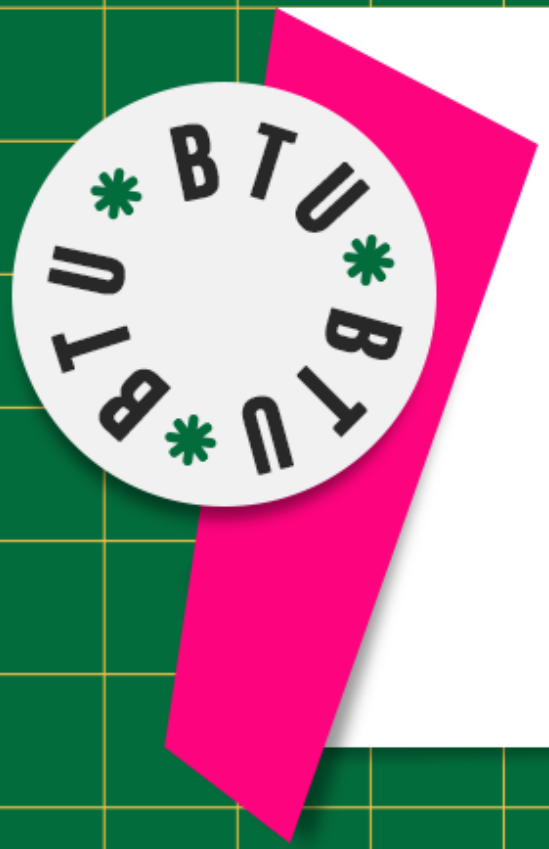
1. AWS Scripted: How to Automate the Deployment of Secure and Resilient Websites with Amazon Web Services VPC, ELB, EC2, RDS, IAM, SES and SNS - კრისტიან ჩერი, 2014წ. , 270 გვერდი
2. Amazon Web Services in Action - ანდრეას ვიტიგი, მიქაელ ვიტიგი, 2016წ., 424 გვ,
3. https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/ec2.html

მადლობა ყურადღებისთვის!

კონტაქტი

თბილისი 0162, საქართველო
ი.ჭავჭავაძის გამზირი 82
INFO@BTU.EDU.GE
032 2 195 015
WWW.BTU.EDU.GE

გამოგვყევით:



ღრუბლოვანი სისტემების (AWS) ავტომატიზაცია Python-ის საშუალებით

Guja Nemsadze

საკითხები

- Amazon Simple Queue Service (SQS)
- SQS features
- SQS Limitations
- Using AWS SQS can be a good decision in
- Amazon Simple Notification Service (SNS)
- Benefits of SNS
- Using AWS SNS can be a good decision in
- Differences of SNS and SQS
- Amazon Simple Email Service (SES)
- Hands On SQS
- SQS with Lambda And Requests Layer

Amazon Simple Queue Service (SQS)

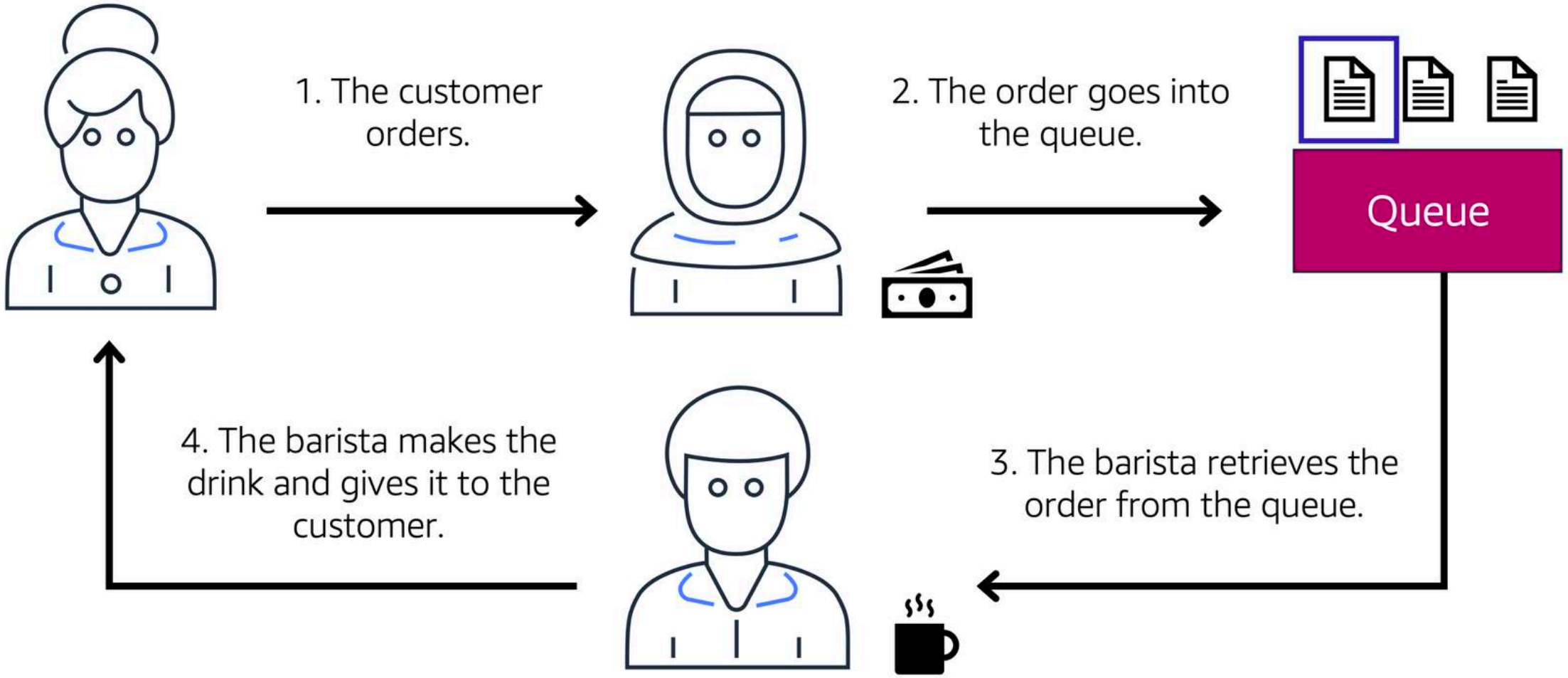
Amazon SQS is a fully-managed **queue service** provided by AWS. Queues are a useful addition to your application architecture with many use cases. They can be used to **throttle load** in a crucial area of your system, reliably process batches of work, or communicate between microservices. There are a number of message queues in popular use today, including RabbitMQ, ActiveMQ, or RocketMQ.

More to Read:
https://prezi.com/dwbwhu_a2pny/amazon-sqs/
https://hevodata.com/learn/aws-sqs

SQS features

- Standard Queues *($0.40 per million requests, or $0.00000040 per request)*
  Standard queues provide best-effort message ordering with at least once delivery. This means that you can expect occasions where messages may arrive **more than once** and, while they usually arrive roughly in the order they are sent, there are **no strict guarantees**.
- FIFO Queues *($0.50 per million requests, or $0.00000050 per request)*
  FIFO queues provide strict ordering guarantees with **exactly once delivery**. With this method, you can be sure about the order of consumption and **avoid duplicate deliveries**. This comes with a tradeoff – the throughput with **FIFO queues is limited** by comparison.
- Dead-letter queues (DLQ)
  SQS allows you to set up a dead-letter queue, where it sends messages that has been unsuccessfully processed a given number of times. This can help you **avoid poisoning your queue** with messages that are **invalid**.
- Long-polling
  SQS uses **HTTP** rather than a stateful connection to the message broker. When your queue is empty, you will need to **continually ping** your queue to see if any new messages have arrived. With long-polling, you can make a request to receive messages and wait up to 20 seconds to see if any messages arrive.

Example: Orders in a queue



SQS Limitations

- There is no limit to the number of messages that can be stored and there is no request limit for **standard queues**!
- The message size is limited to **256KB**. If you have a larger payload, you can store it on **S3** and reference the object key in the message.
- FIFO queues allow you to make 300 requests per second by default. High throughput mode allows you to make 3000 API requests per second. Using batch APIs, you can send and receive up to 30,000 messages per second in a high throughput FIFO queue.
- The number of messages in flight (the state between ReceiveMessage and either DeleteMessage or the visibility timeout) is **120,000** for standard queues and **30,000** for FIFO queues.

Using AWS SQS can be a good decision in

- E-commerce Order Processing
- Video Processing Pipeline
- Email Notification System
- Financial Systems
- etc

Amazon Simple Notification Service (SNS)

It is a web service which makes it easy to set up, operate, and send a notification from the cloud. Was released in 2010, which makes it one of the older AWS services. It provides developers with the highly scalable, cost-effective, and flexible capability to publish messages from an application and sends them to other applications. SNS can also **send** the messages to devices by sending push notifications to **Apple**, Google, **Fire OS**, and **Windows** devices, as well as Android devices in China with Baidu Cloud Push.

Benefits of SNS

- Instantaneous delivery
  SNS is based on **push-based delivery.** This is the key difference between SNS and SQS. SNS is pushed once you publish the message in a topic and the message is delivered to **multiple subscribers**.
- Flexible
  SNS supports multiple endpoint types. Multiple endpoint types can receive the message over multiple transport protocols such as email, SMS, Lambda, Amazon SQS, HTTP, etc.
- Inexpensive
  SNS service is quite inexpensive as it is based on pay-as-you-go model, i.e., you need to pay only when you are using the resources with no up-front costs.

Using AWS SNS can be a good decision in

- E-commerce Order Updates
- System Alerts and Incident Management
- Financial Transaction Alerts
- Service Health Monitoring

Differences of SNS and SQS

- SQS is a **pull-based delivery**, i.e., messages are not pushed to the receivers. Users have to pull the messages from the Queue. SNS is a **push-based** delivery, i.e., messages are pushed to multiple subscribers.
- In SNS service, messages are pushed to the multiple receivers at the same time while in SQS service, messages are not received by the multiple receivers at the same time.
- SQS polling introduces **some latency** in message delivery while SNS pushing pushed the messages to the subscribers **immediately**.

Amazon Simple Email Service (SES)

Amazon SES is a fully-managed email service provided by AWS. It allows for both sending and receiving email.

- Provides Simple Mail Transfer Protocol (SMTP) interface to send emails.
- Allows sending emails in an HTML and text format.
- Accepts messages up to **10MB** in size including any images and attachments that are part of the message. Also customers **can now request** a limit increase to send and receive emails with a message size of up to **40MB**.
- Can send emails to an individual email addresses or a group of recipients.

Hands On SQS

https://replit.com/@JexPY/MortifiedTinyBookmark

SQS with Lambda && Requests Layer

https://cloud.eyedea.cz/api/anonymizer

https://replit.com/@JexPY/AccomplishedPepperyUtilities

ლიტერატურა

1. AWS Scripted: How to Automate the Deployment of Secure and Resilient Websites with Amazon Web Services VPC, ELB, EC2, RDS, IAM, SES and SNS - კრისტიან ჩერი, 2014წ. , 270 გვერდი
2. Amazon Web Services in Action - ანდრეას ვიტიგი, მიქაელ ვიტიგი, 2016წ., 424 გვ,
3. https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/ec2.html

მადლობა ყურადღებისთვის!

კონტაქტი

თბილისი 0162, საქართველო
ი.ჭავჭავაძის გამზირი 82
INFO@BTU.EDU.GE
032 2 195 015
WWW.BTU.EDU.GE

გამოგვყევით: