Building Modern Python Applications on AWS

SENG 4260

Sina Keshvadi
TRU

Part 01 - AWS CLIs and APIs

Introduction to Building Modern Applications

Let’s review the basic AWS knowledge
that you already have.

Amazon Web Services (AWS) Infrastructure

- AWS operates a vast global network of data centers that provides the backbone for its cloud services.
- These data centers are strategically located in various regions around the world, allowing users to deploy resources close to their target audience for improved performance.

Availability Zones

- Availability Zones (AZs) are isolated data centers within a region.
- Each AZ is equipped with independent power, cooling, and networking to ensure fault tolerance.
- They are designed to provide redundancy and high availability, allowing applications to continue running even if one AZ experiences a failure.

Benefits of Availability Zones

- **High Availability**: AZs are physically separate from each other, minimizing the risk of a single point of failure.
- **Fault Isolation**: In the event of a hardware failure or other issue in one AZ, resources in other AZs remain unaffected, maintaining service reliability.
- **Scalability**: Users can distribute their resources across multiple AZs to scale their applications and handle increased workloads.
- **Data Residency and Compliance**: AZs allow users to keep their data and applications within a specific geographic area to compliance with data residency requirements.
- **Disaster Recovery**: By replicating resources across different AZs, businesses can recover quickly from disasters.

AWS Identity and Access Management (IAM)

- **AWS Identity and Access Management (IAM)** is a robust service that enables secure control over AWS resources.
- It allows you to manage users, groups, roles, and permissions to ensure fine-grained access control to AWS services and resources.

Key Components of IAM

- **Users**: Individuals or systems that interact with AWS resources. Each user has a unique set of security credentials for authentication.
- **Groups**: Collections of users with similar permissions. Instead of assigning permissions to individual users, you assign them to groups for easier management.
- **Roles**: Roles are used to delegate permissions to entities like applications or services. They don't have static credentials and are assumed for a specific task.
- **Policies**: IAM policies define the permissions that control what actions are allowed or denied on AWS resources. They can be attached to users, groups, or roles.

Amazon EC2

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable compute capacity in the cloud. It allows users to run virtual servers, known as instances, on-demand.

Key Features of Amazon EC2

- **Scalability**: EC2 instances can be easily scaled up or down to meet changing workload demands.
- **Variety of Instance Types**: EC2 offers a wide range of instance types optimized for different use cases, including compute-optimized, memory-optimized, and storage-optimized.
- **AMI (Amazon Machine Image)**: Users can choose from a selection of pre-configured Amazon Machine Images or create their own.
- **Security Groups and Network Access Control Lists (ACLs)**: These provide fine-grained control over inbound and outbound traffic to instances, enhancing security.
- **Elastic Load Balancing**: EC2 instances can be used in conjunction with Elastic Load Balancers (ELB) to distribute traffic evenly across multiple instances to improve availability and fault tolerance.

Amazon S3 - Simple Storage Service

Amazon Simple Storage Service (Amazon S3) is an object storage service that offers scalable and durable storage for various types of data.

Key Features of Amazon S3

- **Scalability**: S3 provides virtually unlimited storage capacity, allowing users to store and retrieve large amounts of data.
- **Durability and Reliability**: S3 is designed for 99.999999999% (11 9's) durability of objects, making it a highly reliable choice for data storage.
- **Data Management and Versioning**: Users can manage their data through versioning, which enables multiple versions of an object to be stored.
- **Security and Access Control**: S3 supports fine-grained access control policies and offers features like encryption, access logs, and cross-region replication for enhanced security.
- **Lifecycle Policies**: Users can define policies to automatically transition objects to different storage classes or delete them based on specified criteria.

Amazon VPC - Virtual Private Cloud

Amazon Virtual Private Cloud (Amazon VPC) is a networking service that allows users to provision a logically isolated section of the AWS Cloud where they can launch AWS resources.

Key Features of Amazon VPC

- **Subnets**: VPC can be divided into subnets, each residing in a specific Availability Zone. Subnets allow for finer control over resource placement.
- **Route Tables**: VPCs have associated route tables that control the traffic between subnets and the internet. Custom route tables can be created to define specific routing.
- **Internet Gateway**: An Internet Gateway allows communication between instances in a VPC and the internet. It acts as a bridge for inbound and outbound traffic.
- **NAT Gateways**: Network Address Translation (NAT) Gateways allow resources in a private subnet to connect to the internet while remaining secure.
- **Security Groups and NACLs**: These provide security controls at both the instance and subnet levels, allowing for fine-grained access control.

Note:

Building brand new applications on AWS is a different task

than lifting and shifting existing applications into AWS.

We are going to, from scratch,

build a serverless backend

using AWS cloud-native tools and services.

When you have an existing application that you need to move to AWS,

you might first look to using **Amazon EC2** as your virtual machines,

or maybe you might look into using **Docker containers** and
container hosting services like **Amazon Elastic Container Service**,
or **Amazon Elastic Kubernetes Service**.

In modern cloud-native application development,

it's oftentimes the goal to build out serverless architectures

that are scalable, are highly available, and are fully managed.

This means less operational overhead for you and your business,

and more focusing on the applications and business-specific projects

that differentiate you in your marketplace.

This means we will **not** be covering Amazon EC2

or AWS Container services.

Instead we are going to, from scratch,

build a serverless backend

using AWS cloud-native tools and services.

It's commonplace to choose to build API-driven applications.

We first explore how to build an API-driven application using Amazon API Gateway for serverless API hosting.

Then we will add authentication to the API

using Amazon Cognito.

From there, we will add a Lambda backend

that will be triggered by API Gateway.

AWS Lambda for serverless compute

Amazon Cognito for serverless authentication.

Some of the features of our API

will require multiple Lambda functions

to execute in a specific order, like a workflow.

And we will be using AWS Step Functions

to create a serverless workflow.

As you can see, we aren't going to be talking

about front-end application development much in this course.

We are mostly focusing on the backend aspect of this application,

where we'll be standing up an API that a client or front-end can then consume.

In this lecture, we'll be covering how to build

a modern greenfield serverless backend on AWS.

All right, let's dive in.

Main Application

In the exercises, you will start by installing and configuring the **AWS CLI**,

installing the **AWS SDK**, exploring the **source code**, and setting up **AWS resources** that the application will use.

A frontend website (that you can use to interact with the backend API) is provided.

Then, you will begin to build the backend API with **Amazon API Gateway**, add **authentication** to that API, create the backend compute functions with **AWS Lambda**, and create an asynchronous workflow with **AWS Step Functions**. You will also implement distributed tracing with **AWS X-Ray**, use monitoring features, and improve performance for the distributed application.

AWS services used:

- Amazon Simple Storage Service (Amazon S3)
- Amazon API Gateway
- Amazon Cognito
- AWS Lambda
- AWS Step Functions
- AWS X-Ray
- AWS Systems Manager Parameter Store.

Demo Code

The demo code you will need for this lecture can be found in Moodle alongside this lecture.

Please download it now as you will refer to it later in the lecture.

Architecture for the Cloud

Throughout this course,
we're going to be building a backend API
to demonstrate how to use a variety of AWS services
and show how they integrate and work together.

What we're going to build is an API
that allows you to query data being stored in Amazon S3.

The data being stored in S3 is a record of dragons being sighted all around the world.

We have a JSON file that contains information about each dragon that we are aware of.

Dragons.json


```
{
        "description_str":"From the northern
fire tribe, Atlas  was born from the ashes
of his fallen father in combat. He is
fearless and does not fear battle.",
        "dragon_name_str":"Atlas",
        "family_str":"red",
        "location_city_str":"anchorage",
        "location_country_str":"usa",
        "location_neighborhood_str":"w
fireweed ln",
         "location_state_str":"alaska"
      }
```

So we have this data in JSON
and people want to query the data and consume it.

We are going to design a system to make that possible.

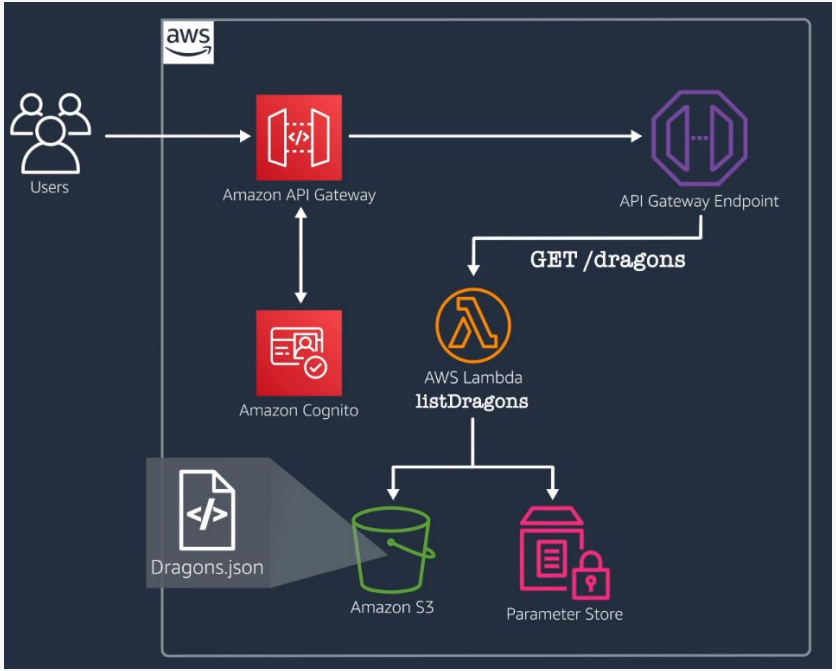
Now, for various security and design reasons
you generally don't want to expose your data directly to your consumers.

Instead, you want to expose your data to your consumers via
an API, or application programming interface.

We are going to build out an API using
Amazon API Gateway to expose that data securely.

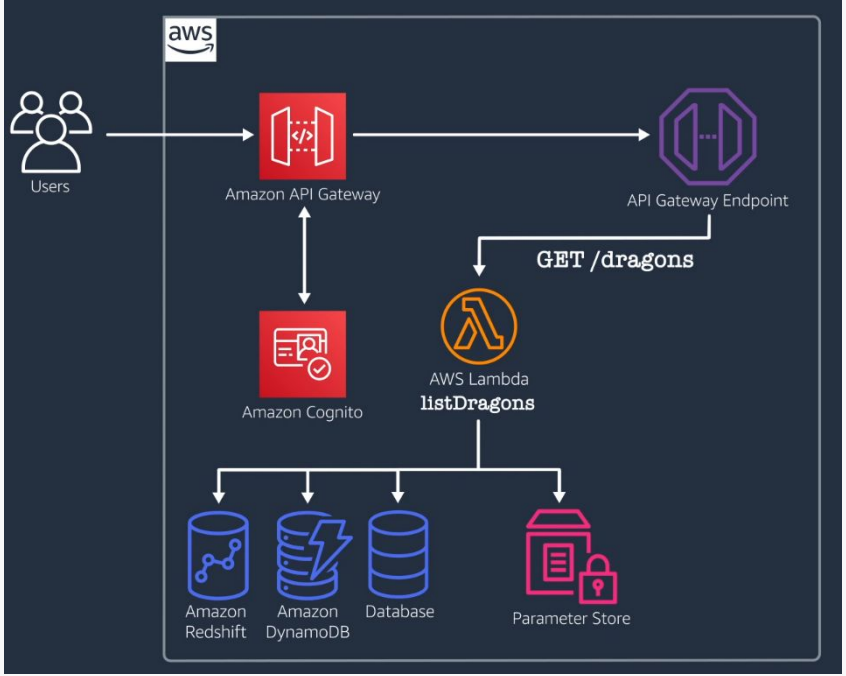
Knowing that our data is being stored in S3
and we will be using API Gateway as a front door to our backend,
we will create an end point for these `/dragons` resource in API Gateway.

You will be able to get a list of all dragons
using the GET HTTP method on the dragons resource.



API Gateway will accept the requests,
perform authorization using **Amazon Cognito**, validate the payload,
and if all that passes, it will invoke the backend to query the dragon data.

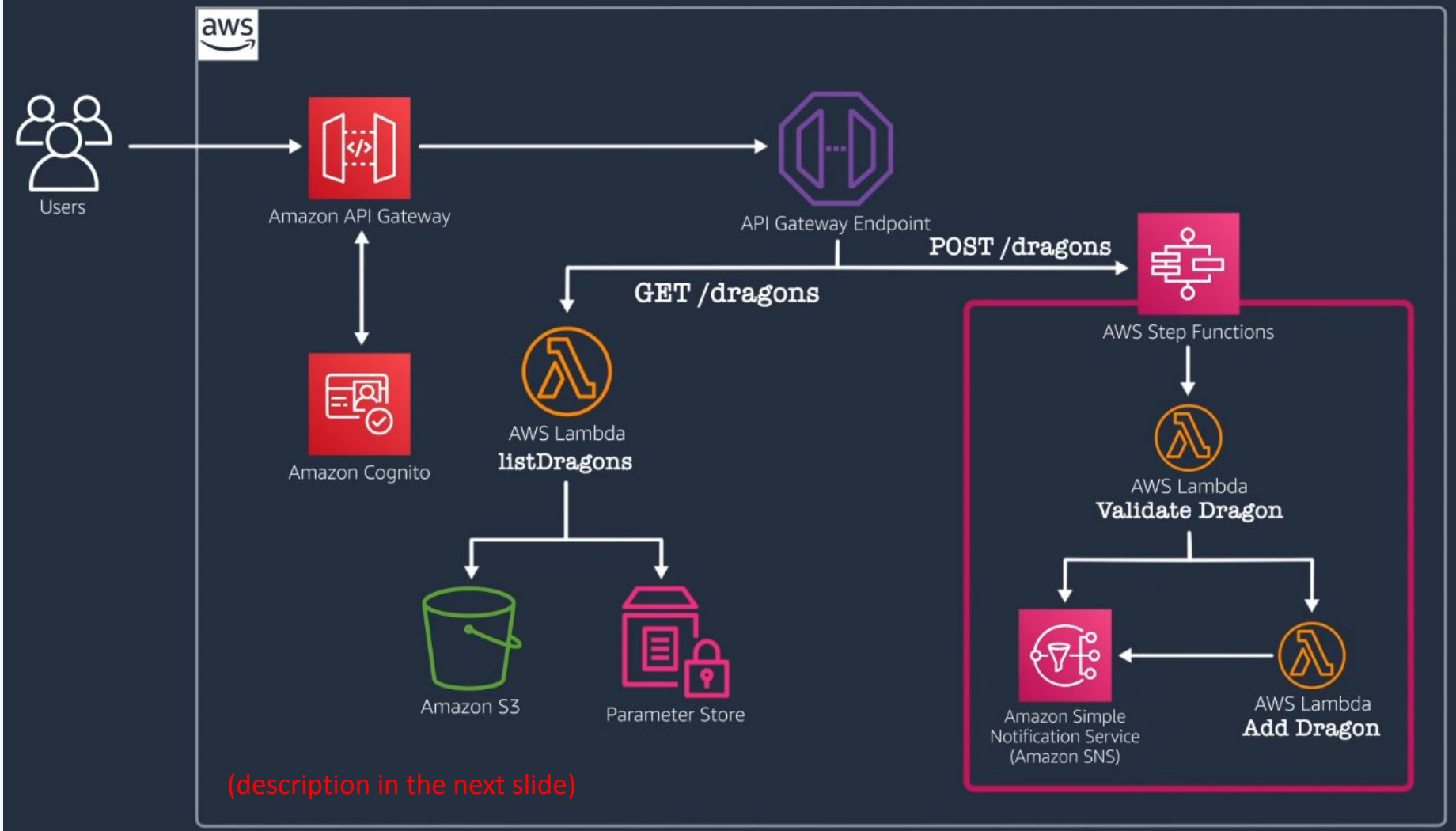
That backend for the GET request will be hosted by **AWS Lambda**.
The Lambda function will be reading the JSON data file in S3 using the API for S3 Select.



In other situations, you might be using a database to host this data, or maybe aggregating data from multiple places.

But for us, we will just be using S3 though it could always be converted to using a database later on if you wish.

The other feature our API supports
is we will allow people to report new dragon sightings.



(description in the next slide)

The way users will interact with this feature is by sending a **POST HTTP request** to the /dragons resource in API Gateway.

For the report dragon feature, we will be using **AWS Step Functions**, AWS Lambda and Amazon Simple Notification Service, or SNS, to create an **asynchronous** dragon reporting system.

This reporting process is asynchronous because we need to do some backend data validation that could take some time and we don't want people waiting on an immediate response.

Instead, we alert the user when the process has completed.

This is a good example to show you how to set up a workflow that would kick off when a request hits your API, then your backend would do whatever work needs to be done and return the response when it's ready via SNS.

This is known as "**request offloading**" and it allows you to respond to your client without the entire process having finished yet.

Introduction to AWS Cloud 9

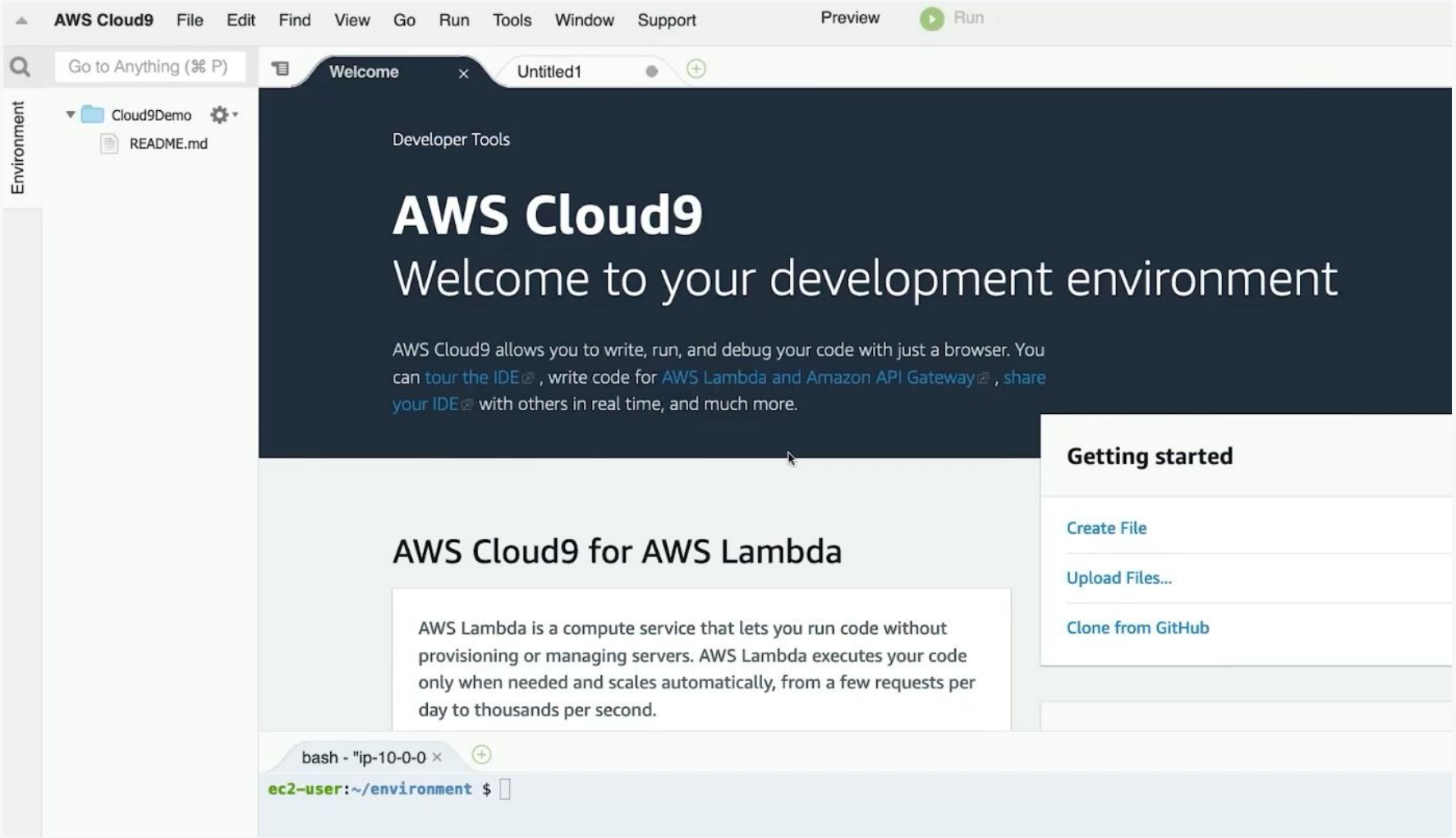
AWS Cloud9 is a cloud-based integrated development environment, or IDE, that lets you write, run, and debug your code within your browser.

It includes a code editor, debugger, and terminal.

Cloud9 comes pre-packaged with essential tools for popular programming languages,
including JavaScript, Python, PHP and more,
so you don't need to install files or configure your development machine to start new projects.

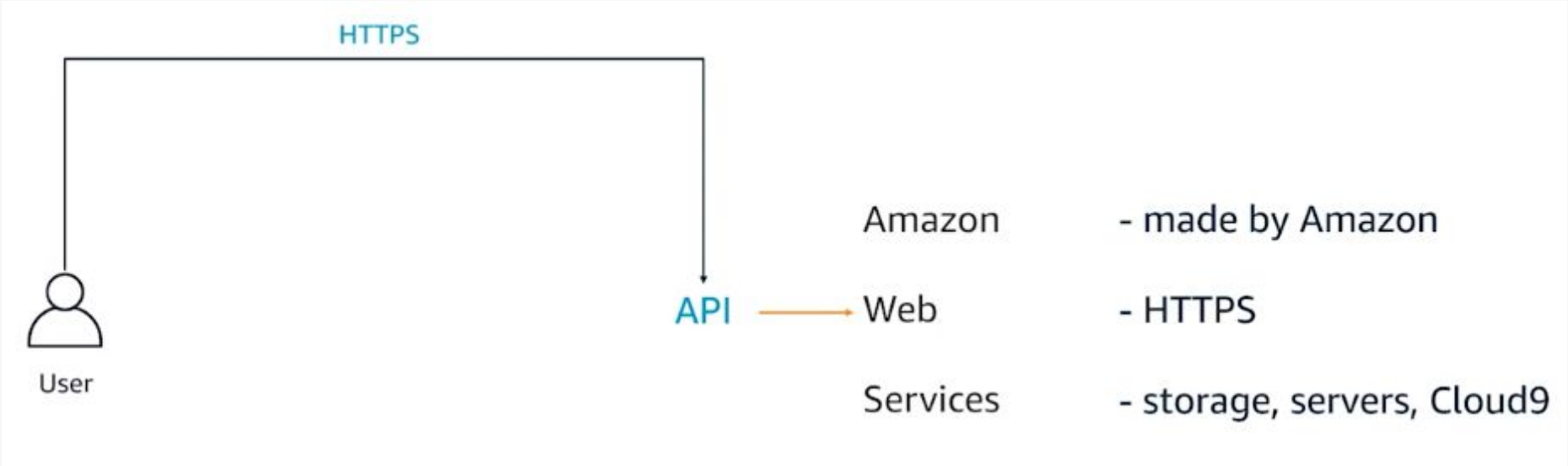
With Cloud9, you can quickly share your development environment with your team, enabling you to pair programs and track each other's inputs in real time.

One of the other major benefits of Cloud9 is the ability to work with the AWS command line tools with little to no setup or configuration of the tools themselves.


AWS Cloud9 File Edit Find View Go Run Tools Window Support
Preview
Run
Go to Anything (⌘ P)
Welcome
Untitled1
Environment
Cloud9Demo
README.md
Developer Tools
AWS Cloud9
Welcome to your development environment
AWS Cloud9 allows you to write, run, and debug your code with just a browser. You can tour the IDE, write code for AWS Lambda and Amazon API Gateway, share your IDE with others in real time, and much more.
Getting started
Create File
Upload Files...
Clone from GitHub
AWS Cloud9 for AWS Lambda
AWS Lambda is a compute service that lets you run code without provisioning or managing servers. AWS Lambda executes your code only when needed and scales automatically, from a few requests per day to thousands per second.
bash - "ip-10-0-0
ec2-user:~/environment $

Take the time to dive through Cloud9
to figure out how it would best work for you.

Introduction to AWS API Management Console CLI SDK



Each of AWS services have an API that is made available to you.

Those APIs are the entry point of AWS and to interact with them, and a user has to make an HTTPS request to that API.

Let's take a look at how you could do that.
Let's say that you wanted to create a new S3 bucket.

The first thing you will do is browse the documentation at aws.amazon.com.

Then click on Amazon S3 in the list, you will click on API reference.

Then you will expand S3 reference,
expand the actions,
expand Amazon Simple Storage Service.

And then there it is, select CreateBucket.

CreateBucket

PDF

Creates a new S3 bucket. To create a bucket, you must register with Amazon S3 and have a valid AWS Access Key ID to authenticate requests. Anonymous requests are never allowed to create buckets. By creating the bucket, you become the bucket owner.

Not every string is an acceptable bucket name. For information about bucket naming restrictions, see Bucket naming rules.

If you want to create an Amazon S3 on Outposts bucket, see Create Bucket.

By default, the bucket is created in the US East (N. Virginia) Region. You can optionally specify a Region in the request body. To constrain the bucket creation to a specific Region, you can use `LocationConstraint` condition key. You might choose a Region to optimize latency, minimize costs, or address regulatory requirements. For example, if you reside in Europe, you will probably find it advantageous to create buckets in the Europe (Ireland) Region. For more information, see Accessing a bucket.

> **Note**
> If you send your create bucket request to the `s3.amazonaws.com` endpoint, the request goes to the `us-east-1` Region. Accordingly, the signature calculations in Signature Version 4 must use `us-east-1` as the Region, even if the location constraint in the request specifies another Region where the bucket is to be created. If you create a bucket in a Region other than US East (N. Virginia), your application must be able to handle 307 redirect. For more information, see Virtual hosting of buckets.

Permissions

In addition to `s3:CreateBucket`, the following permissions are required when your `CreateBucket` request includes specific headers:

- **Access control lists (ACLs) -** If your `CreateBucket` request specifies access control list (ACL) permissions and the ACL is public-read, public-read-write, authenticated-read, or if you specify access permissions explicitly through any other ACL, both `s3:CreateBucket` and `s3:PutBucketAcl` permissions are needed. If the ACL for the `CreateBucket` request is private or if the request doesn't specify any ACLs, only `s3:CreateBucket` permission is needed.

You can find every details for each of the APIs of AWS

by following the exact same path.

we need to do an HTTPS request
that needs to be sent to AWS.

The following operations are related to `CreateBucket`:

- PutObject
- DeleteBucket

Request Syntax

```
PUT / HTTP/1.1
Host: Bucket.s3.amazonaws.com
x-amz-acl: ACL
x-amz-grant-full-control: GrantFullControl
x-amz-grant-read: GrantRead
x-amz-grant-read-acp: GrantReadACP
x-amz-grant-write: GrantWrite
x-amz-grant-write-acp: GrantWriteACP
x-amz-bucket-object-lock-enabled: ObjectLockEnabledForBucket
x-amz-object-ownership: ObjectOwnership
<?xml version="1.0" encoding="UTF-8"?>
<CreateBucketConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
   <LocationConstraint>string</LocationConstraint>
</CreateBucketConfiguration>
```

URI Request Parameters

The request uses the following URI parameters.

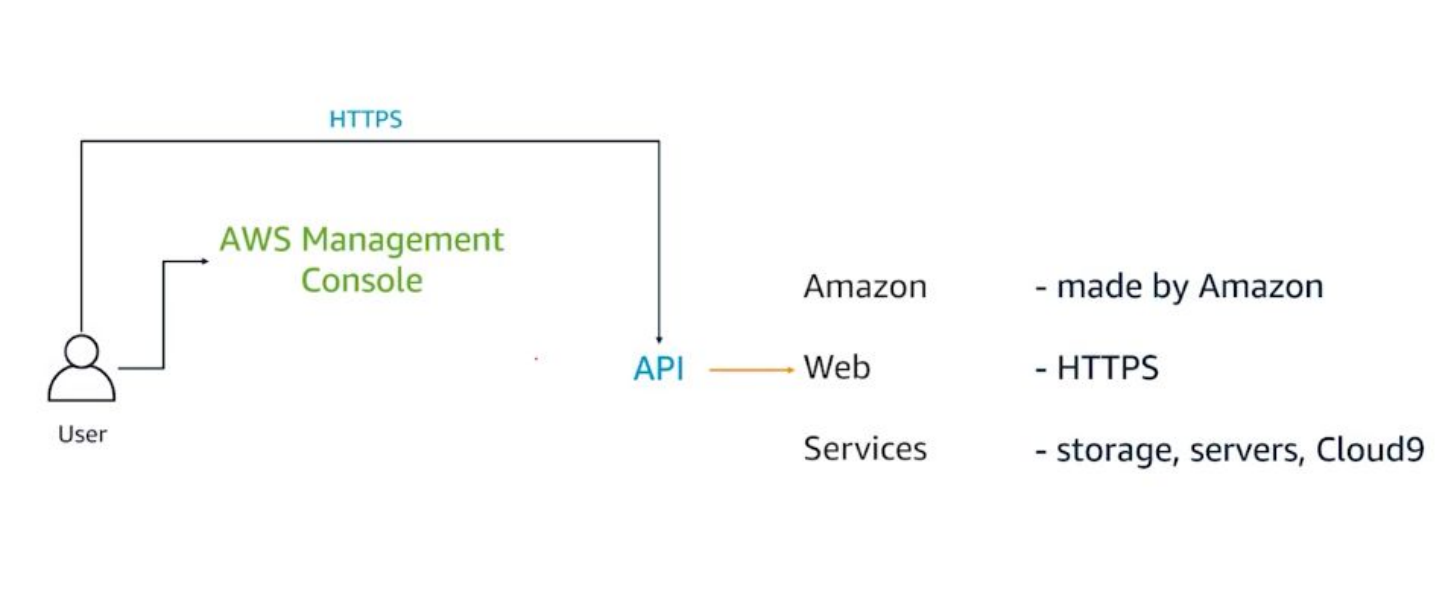
Bucket

So you will need to craft this PUT request and
specify all of the parameters that you see inside of that page.

Whoa, that sounds pretty difficult to do, doesn't it?

In fact, doing everything over the API
is the hardest way you can interact with AWS.

The easiest way is via a graphical user interface
that is available over the web that
we call the AWS Management Console.



AWS Management Console interacts with the AWS APIs directly.

This graphical user interface sounds easy enough
to work with when you, as a user, want to interact with it.

However, it becomes harder to work with when you want to repeat a task.

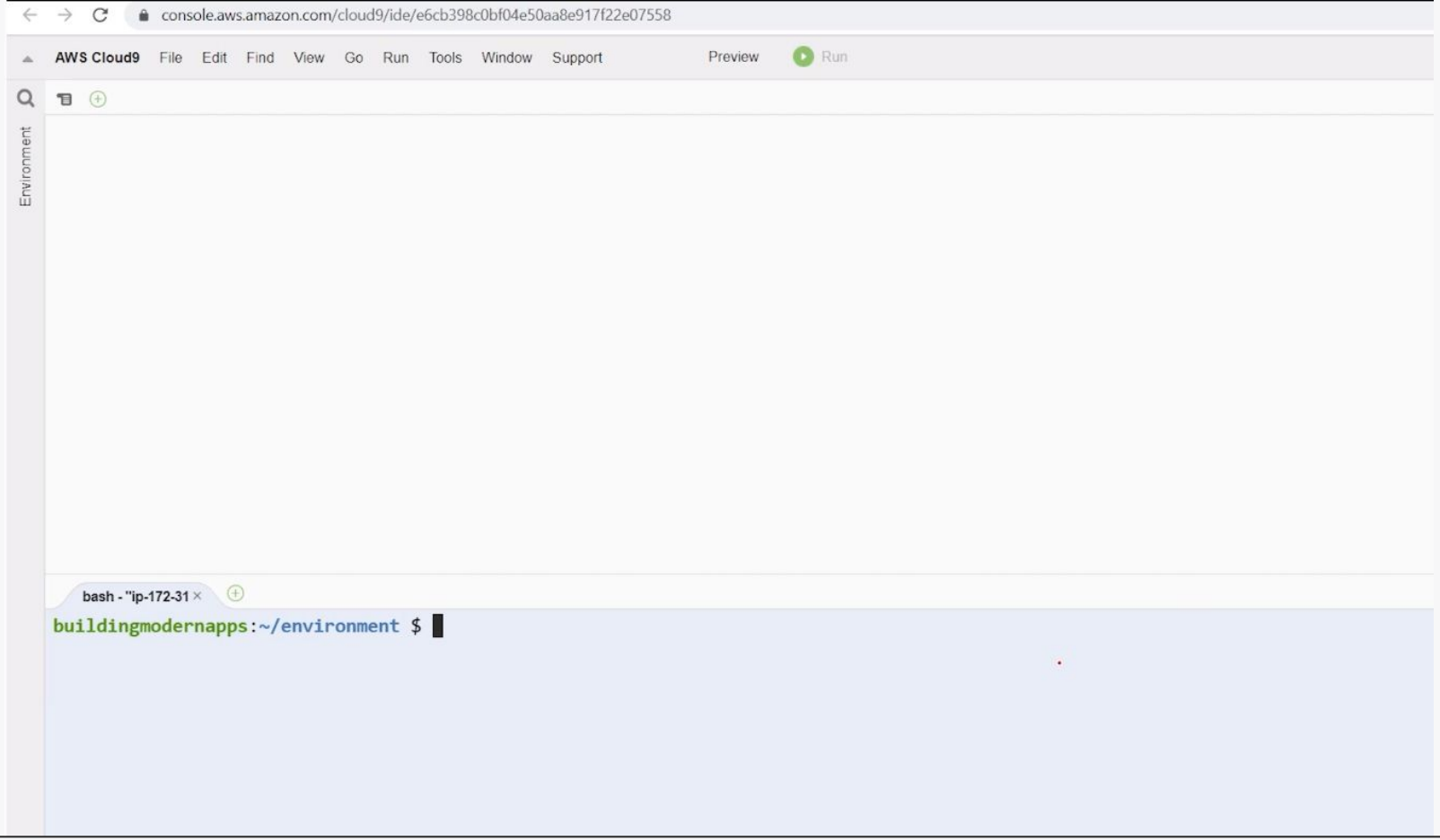
For example, let's say you wanted to create a few buckets in different regions.

Now, you will have to change your region, click on 'create bucket' again, fill everything like you did before, and really remember everything that you've done and then not mess up in between all of those tries.

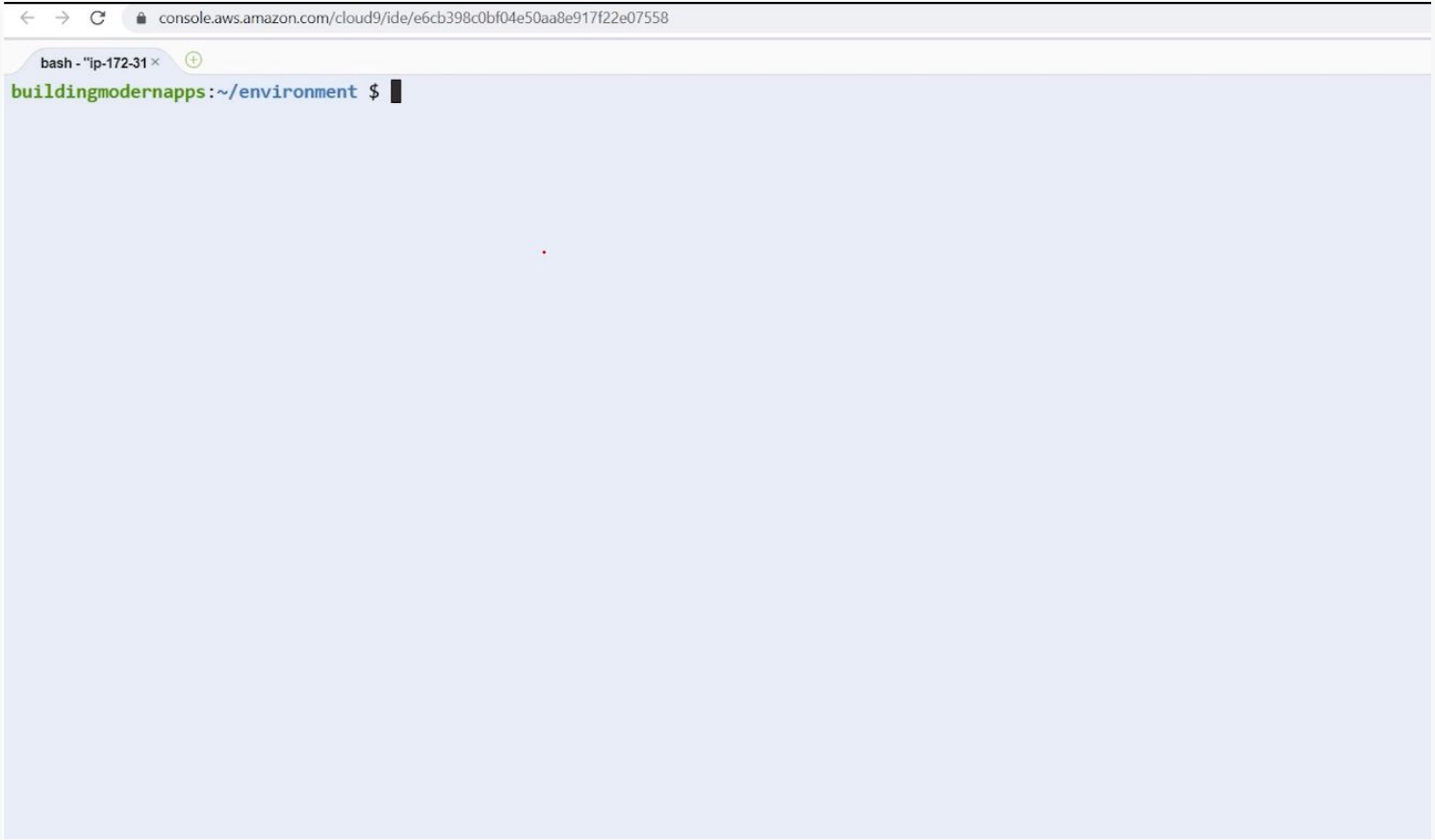
The better option here is for you to use

the AWS Command Line Interface, or CLI,

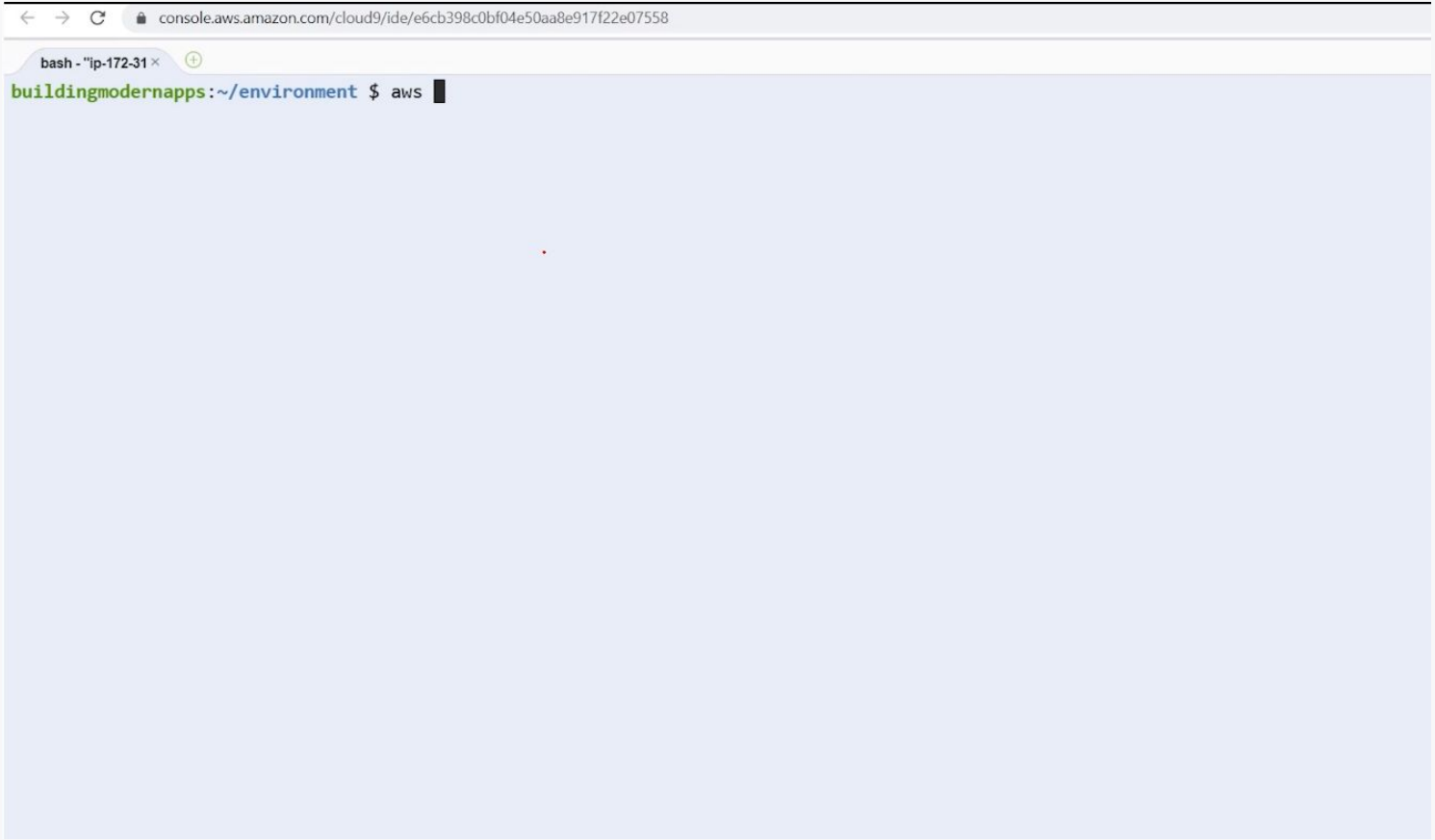
which is like the AWS Management Console, it uses the API of AWS behind the scenes.



Let's take a look, by going into our Cloud9 environment that is already set up.



cloud9 is tab based.



To interact with the CLI, you will use the **AWS** command. For example, type 'AWS help'

```
buildingmodernapps:~/environment $ aws help
buildingmodernapps:~/environment $
```

```
AWS()                                                                AWS()



NAME
       aws -

DESCRIPTION
       The  AWS  Command  Line  Interface is a unified tool to manage your AWS
       services.

SYNOPSIS
          aws [options] <command> <subcommand> [parameters]
```

```
$ aws help
```

```
buildingmodernapps:~/environment $ aws help
buildingmodernapps:~/environment $ aws help
buildingmodernapps:~/environment $ aws s3 help
```

$ aws s3 help

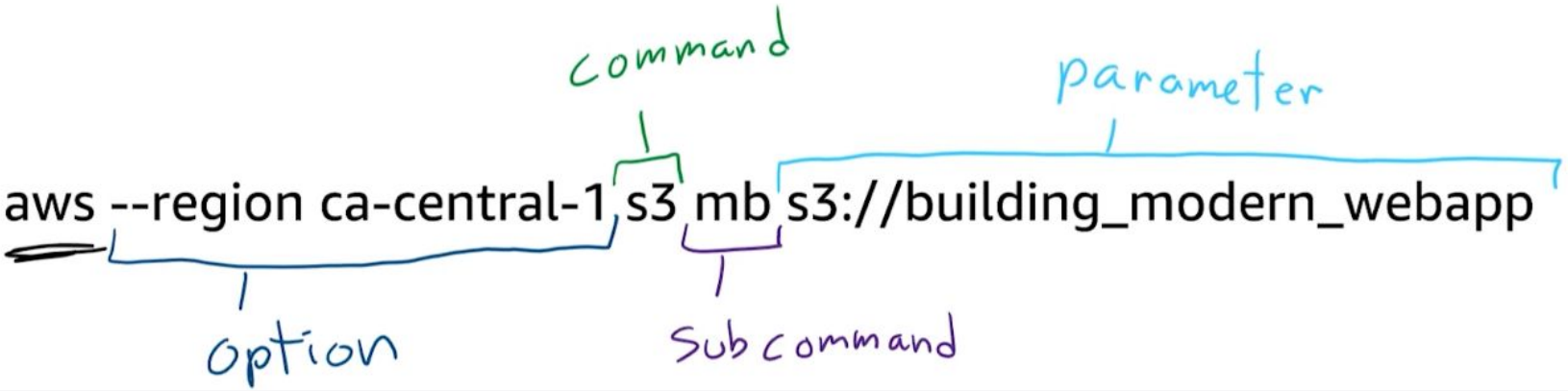
```
aws --region ca-central-1 s3 mb s3://building_modern_webapp
```

- `-region ca-central-1` (option)

`S3` is the command

`mb` is the sub command (mb stands for make bucket)

`S3://building-modern-webapp` is the parameter


command
parameter
aws --region ca-central-1 s3 mb s3://building_modern_webapp
option
Sub command

```
buildingmodernapps:~/environment $ aws help
buildingmodernapps:~/environment $ aws help
buildingmodernapps:~/environment $ aws s3 help
buildingmodernapps:~/environment $ aws s3 mb help
```

Another way is to look at AWS Command Line Interface in

https://docs.aws.amazon.com/

AWS Command Line Interface Documentation

The AWS Command Line Interface (AWS CLI) is a unified tool that provides a consistent interface for interacting with all parts of Amazon Web Services. AWS CLI commands for different services are covered in the accompanying user guide, including descriptions, syntax, and usage examples.

Latest version

User Guide

Describes all the AWS CLI concepts and provides instructions on using the various features of the latest version of the AWS CLI. Version 2 contains all the latest and adds new features.

HTML | PDF | GitHub

Command Reference

Describes the latest version of the AWS CLI in detail and provides basic syntax, options, and usage examples for each operation.

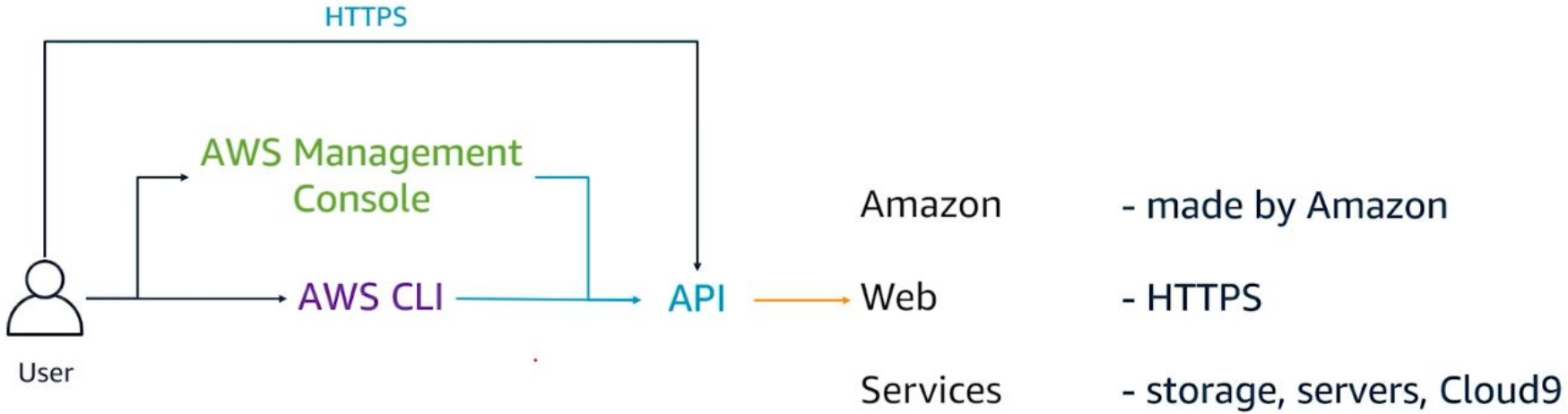
AWS SDKs and Tools Reference Guide

AWS SDKs and Tools shared information including shared config and credential files, configuration and authentication settings, AWS Common Runtime (CRT) libraries, and maintenance and support.

HTML | PDF

AWS Code Examples Repository

The AWS Code Examples Repository contains

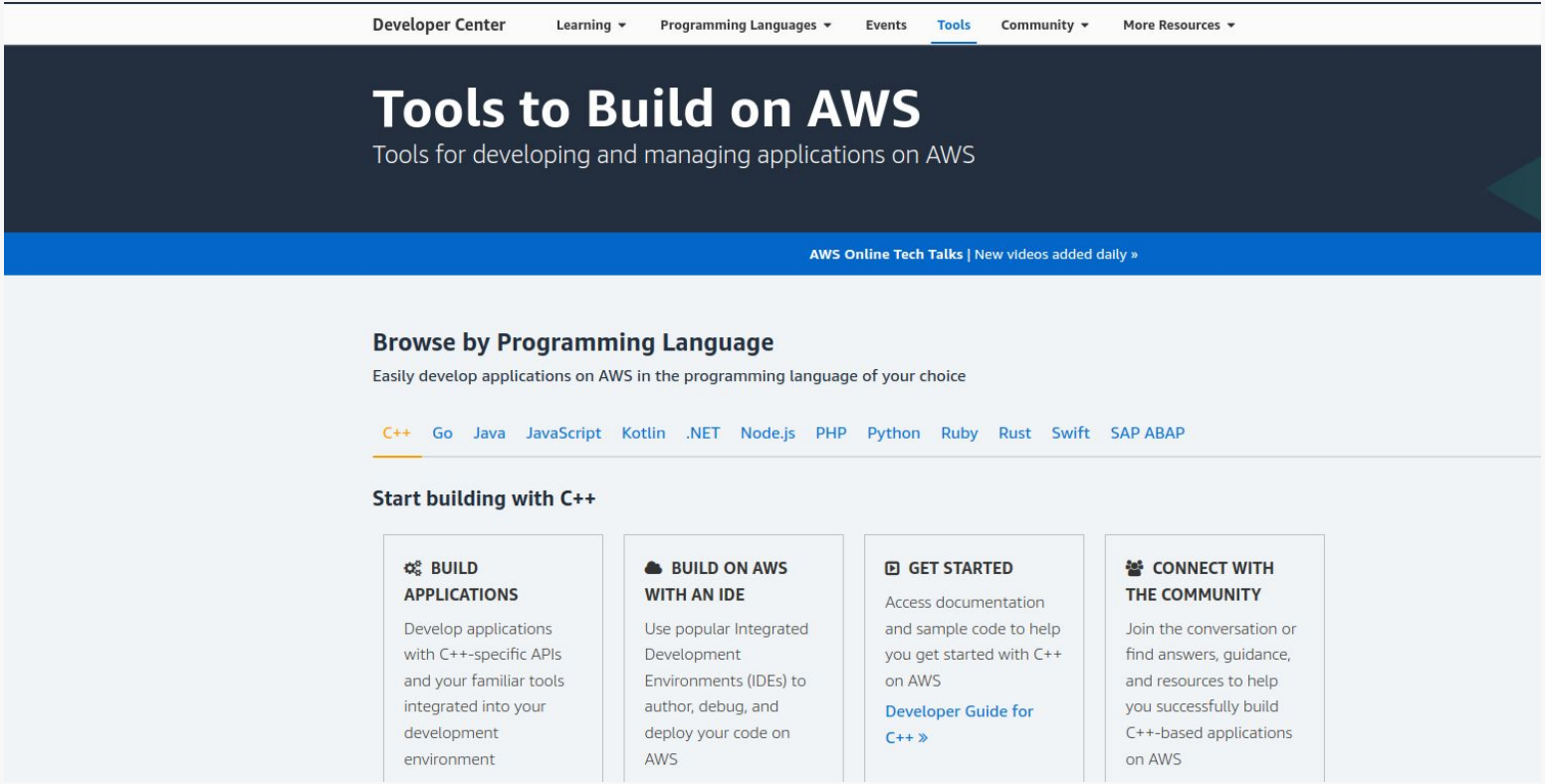


At this point, we saw three ways to interact with AWS:

the Management Console, the AWS CLI, as well as one that you probably shouldn't be directly be interacting with, which is the API itself.

However, none of these will be used by code that you will write.

Instead, you should use
the AWS Software Development Kit or SDK
for the language that you are using.



https://aws.amazon.com/developer/tools/

Recap:

There are four ways to interact with AWS:

1. Directly use the API (not recommended)
2. Management Console, for when you start
3. CLI, for repeatable tasks
4. SDK for your code to interact with AWS.

All of those uses the API of AWS to reach the services of AWS.

AWS CLI Configuration

Now that you have gotten an introduction about APIs,

we're going to go into a little detail about CLI in which you can access APIs in AWS.

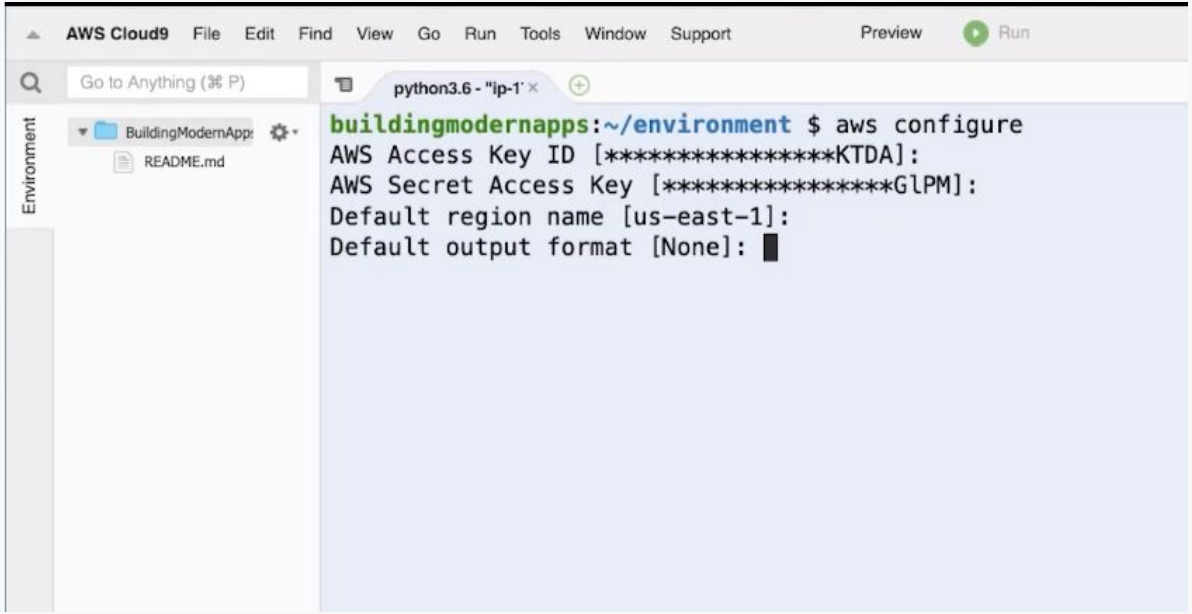
First, it cannot be stressed enough that
almost everything you do in AWS is done as an API call.

Second, the AWS CLI (command line interface) is one of
the easiest and most direct ways to access the API calls that you need to make.

to use CLI, you must configure AWS

simply run:

```
$ aws configure
```



answer these questions for configuration

AWS SDK Exploration (Python)
(boto3)

In this section, we learn
how to install and use the AWS SDK for Python,
otherwise known as boto3.

As you already know, everything in AWS is an API call.

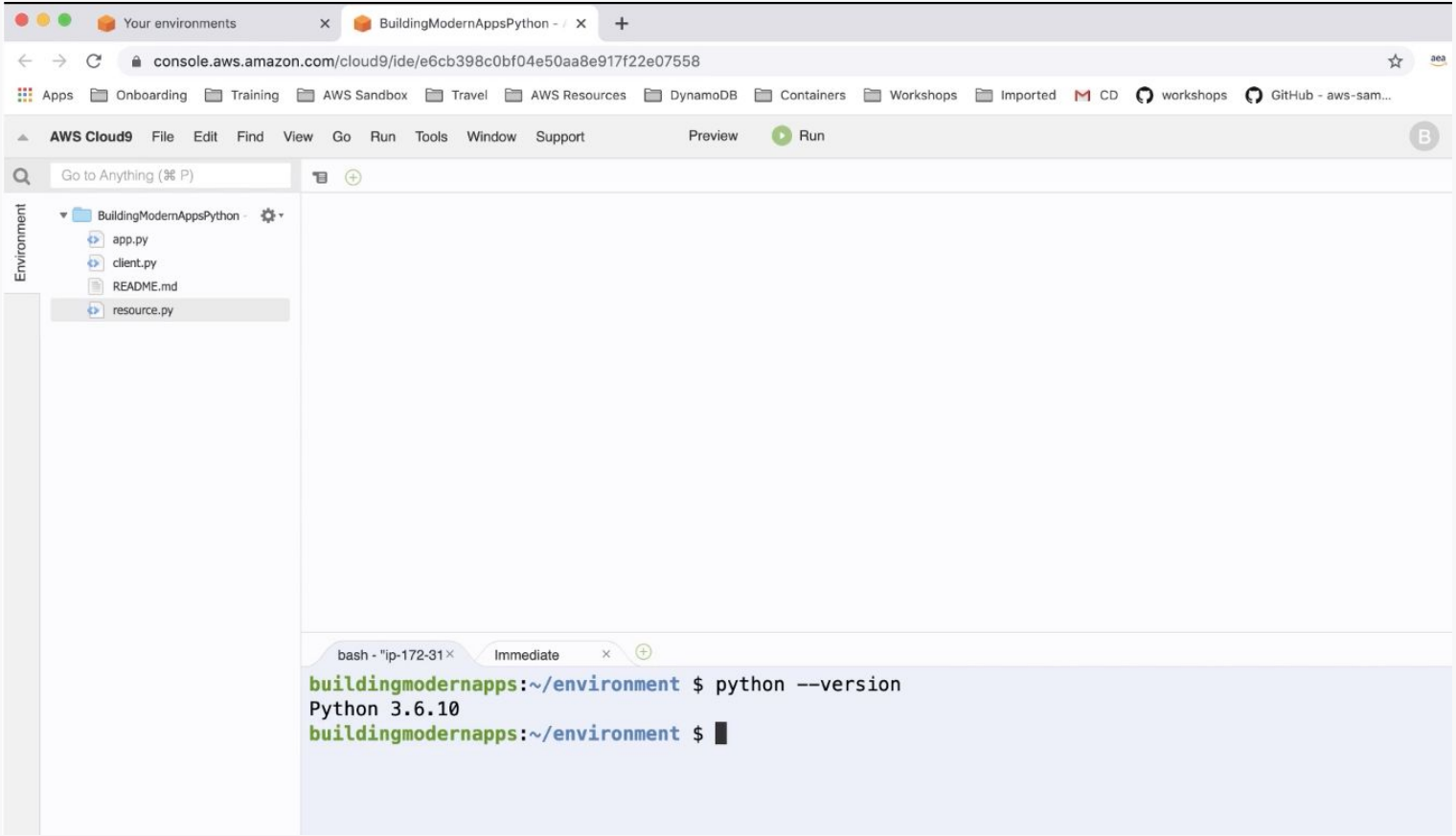
Each different service has its own set of APIs.

And you will invoke these APIs to interact with AWS resources.

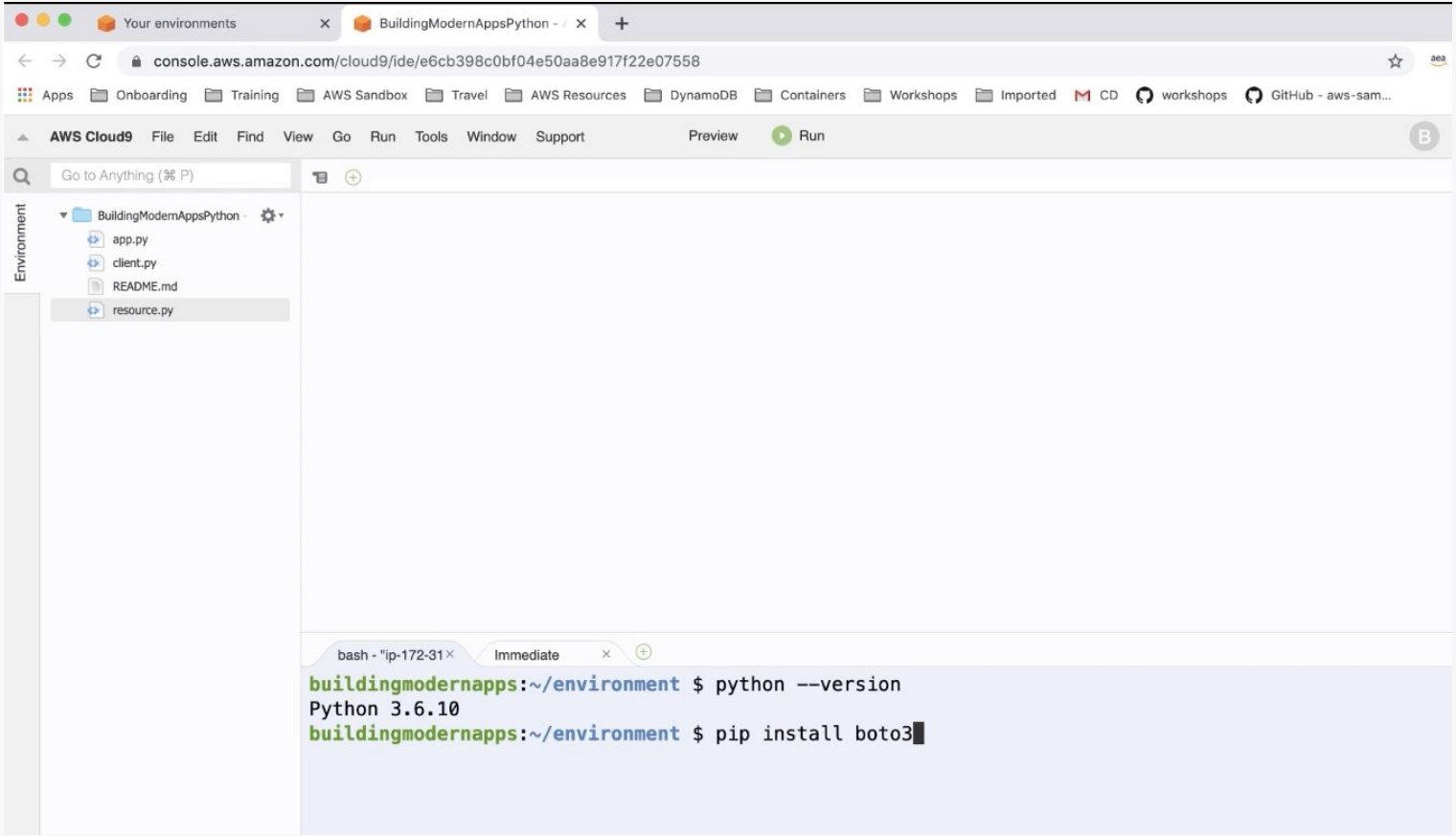
Every API call that is made must be signed and authenticated in order to be successful.

AWS SDK handles much of the complexity involved in interacting with AWS APIs.

Let's take a look at how to set up boto3 using Cloud9.



Note: I'm already logged into this Cloud9 instance.
Before anything, check the version of Python installed on the instance.



Note: if you were to see the word boto3 anywhere, just know that that's the AWS SDK.

So before we dive into our dragon application code,
let's take a few minutes to discuss the two ways
you can interact with AWS APIs using boto3.

In boto3 you can use the client or the resource
to interact with AWS APIs.

The client is a low level interface to AWS,
whose methods map close to one to one with the service APIs.

And it requires some more granular knowledge of the API.

But it also is one to one, so you can get at any API that you may need.

The low level client will return a response to you as a Python dictionary.

It is up to you to traverse or otherwise process the response for the data that you need.

So you may need to tinker with it a little bit to get the data that you want.

The other way to interact with AWS and boto3 is a resource.

The resource is a higher level mapping to the AWS API,
that is object oriented in nature.

It's really a wrapper around the low level API, trying to make your life much easier.

However, resources only expose a subset of AWS APIs,
and therefore you *cannot* do *everything* with a resource.

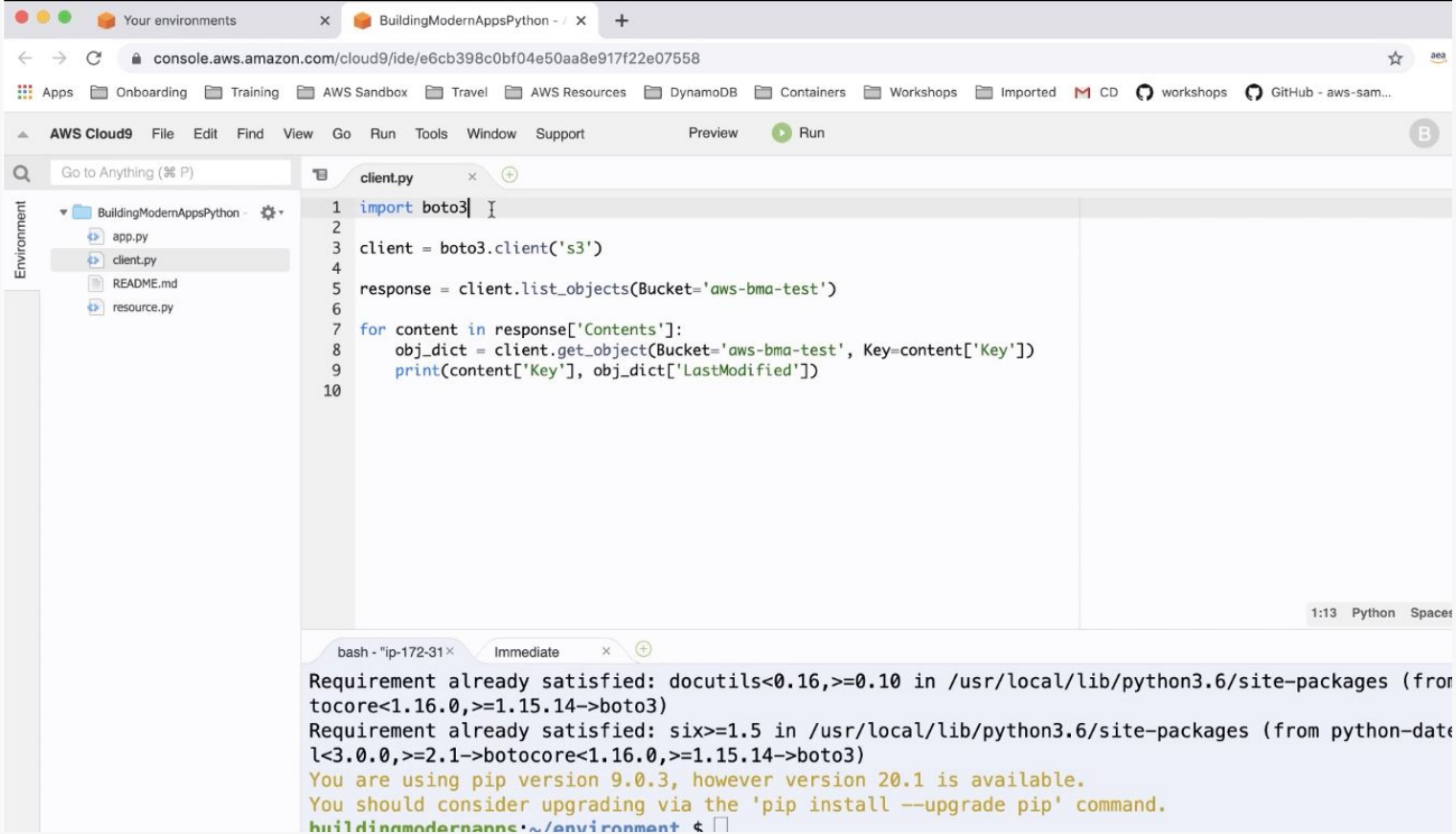
Resources do however, return to you collections of objects and
handle pagination, where the client doesn't.

Code using the resource can be much simpler and easier to read and write,
so if you can use it, go ahead and use the resource.

Nothing prevents you from using both,
as the client interface is actually available within the resource interface.

Once you have a client or resource,
you can then invoke the method on that object which will then call the AWS API for you.

Let's take a look at a simple example of listing all the objects in an S3 bucket using first a client and then a resource.



import boto3 allows to have access to the AWS SDK throughout the script.

```
import boto3

client = boto3.client("s3")

response = client.list_objecs(Bucket="aws-bma-test")

for content in response["Contents"]:
    obj_dict = client.get_object(Bucket="aws-bma-test", Key=content["Key"])
    print(content["Key"], obj_dict["LastModified"])
```

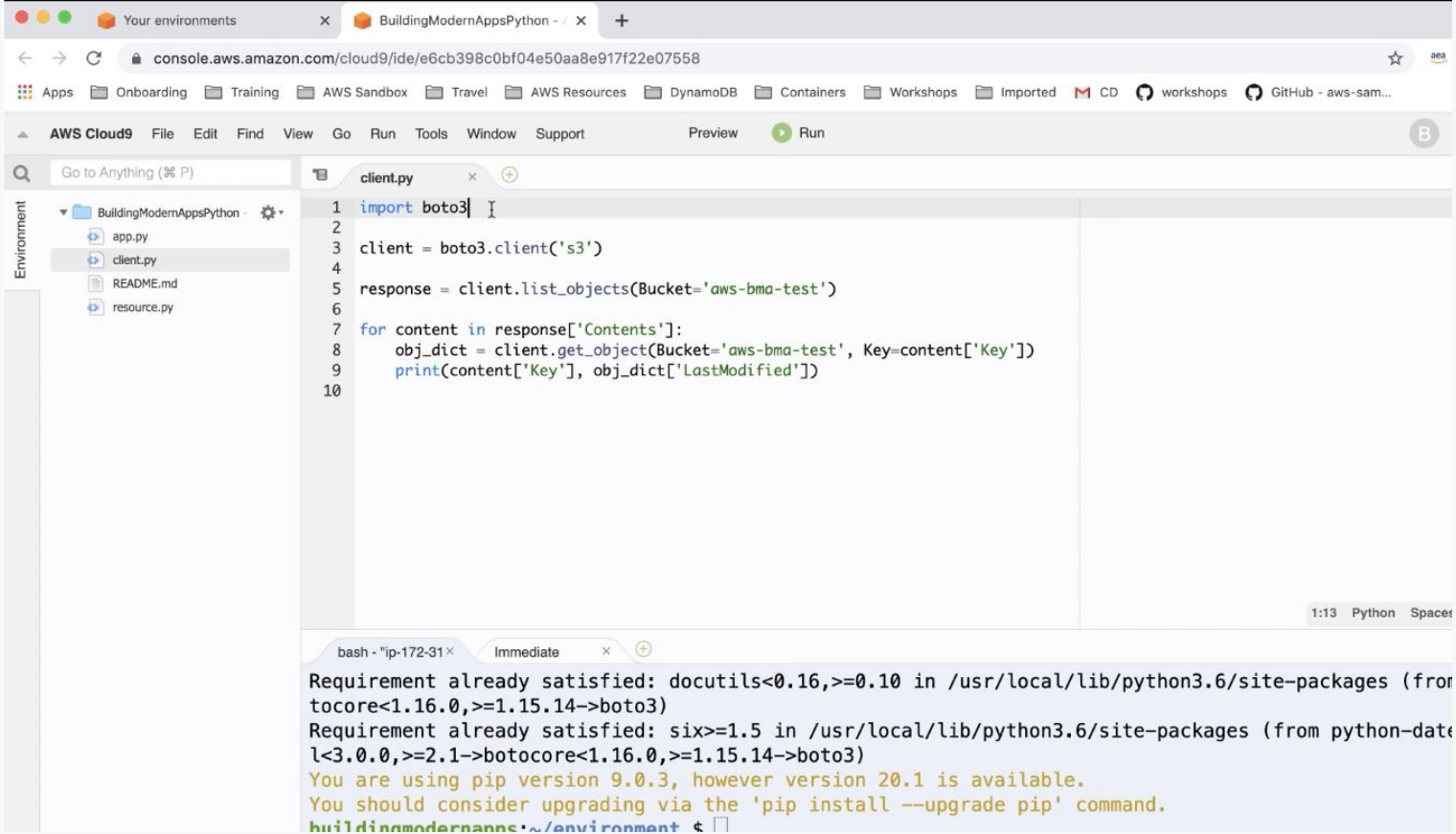
Next, creating the client object by calling boto3.client, passing in the name of the service that you want the client for.

```
import boto3

client = boto3.client("s3")

response = client.list_objecs(Bucket="aws-bma-test")

for content in response["Contents"]:
    obj_dict = client.get_object(Bucket="aws-bma-test", Key=content["Key"])
    print(content["Key"], obj_dict["LastModified"])
```

Then call *client.list_objects* is an API from S3, and passing in this bucket name, aws-bma-test. This is just a bucket that I already have in my account. It's just a test bucket and then I'm storing that in a response.
Again, that's going to come back as a Python dictionary.

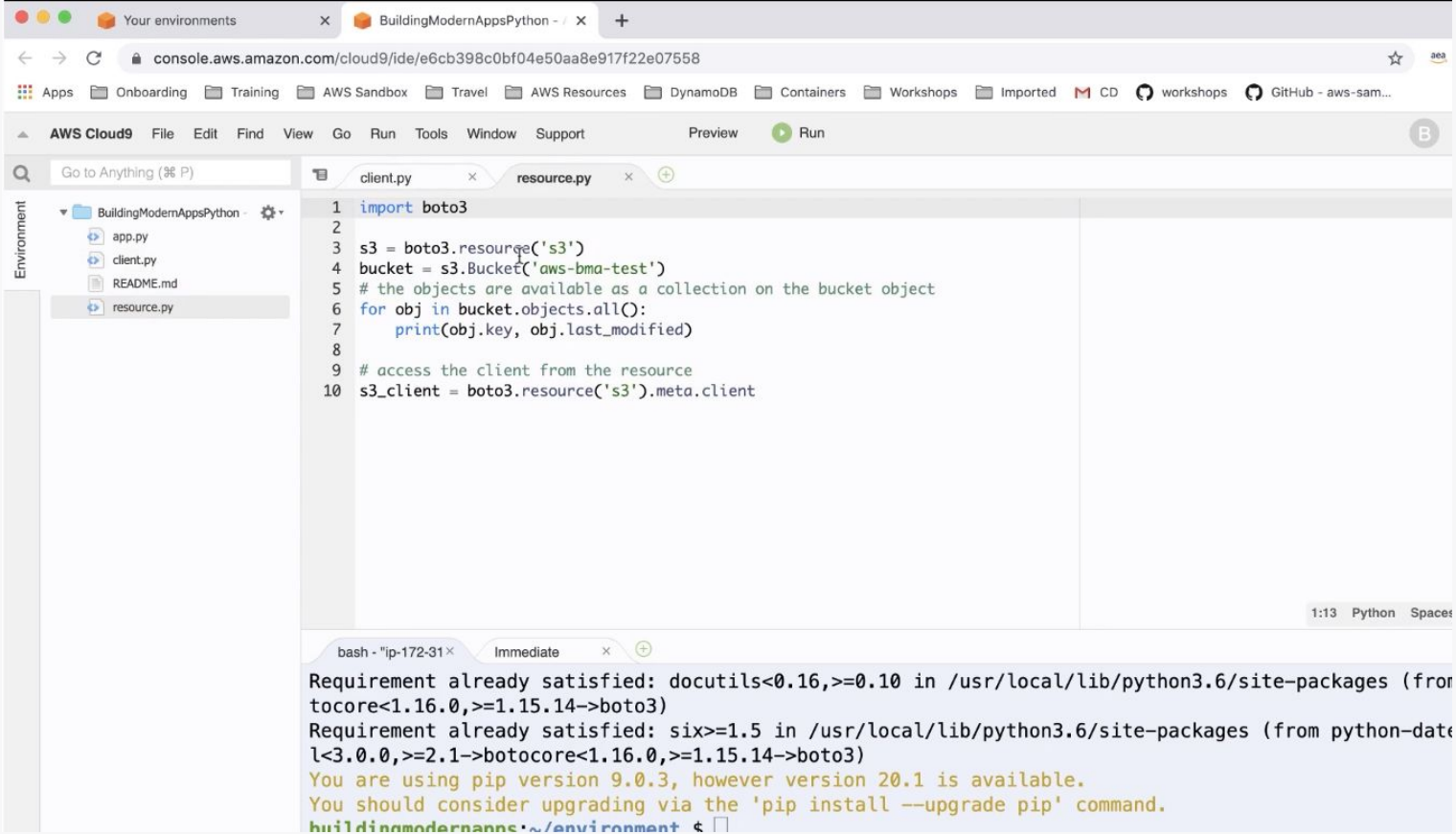
```
import boto3

client = boto3.client("s3")

response = client.list_objecs(Bucket="aws-bma-test")

for content in response["Contents"]:
    obj_dict = client.get_object(Bucket="aws-bma-test", Key=content["Key"])
    print(content["Key"], obj_dict["LastModified"])
```

Next I'm actually iterating over all the contents that came back in that response. And I am then calling the get_object method off of the S3 client for each individual object in that bucket, providing the key and the bucket. And then I'm printing out the key name as well as the last modified date for those objects.



So this is how we would list all of the objects in that S3 bucket using the client.

Now let's go ahead and look at how
we do that for the resource.


Your environments
BuildingModernAppsPython -
console.aws.amazon.com/cloud9/ide/e6cb398c0bf04e50aa8e917f22e07558
Apps
Onboarding
Training
AWS Sandbox
Travel
AWS Resources
DynamoDB
Containers
Workshops
Imported
CD
workshops
GitHub - aws-sam...
AWS Cloud9
File
Edit
Find
View
Go
Run
Tools
Window
Support
Preview
Run
Go to Anything (⌘ P)
Environment
BuildingModernAppsPython
app.py
client.py
README.md
resource.py
client.py
resource.py
1 import boto3
2
3 s3 = boto3.resource('s3')
4 bucket = s3.Bucket('aws-bma-test')
5 # the objects are available as a collection on the bucket object
6 for obj in bucket.objects.all():
7 print(obj.key, obj.last_modified)
8
9 # access the client from the resource
10 s3_client = boto3.resource('s3').meta.client
1:13
Python
Spaces
bash - "ip-172-31
Immediate
Requirement already satisfied: docutils<0.16,>=0.10 in /usr/local/lib/python3.6/site-packages (from
botocore<1.16.0,>=1.15.14->boto3)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.6/site-packages (from python-date
l<3.0.0,>=2.1->botocore<1.16.0,>=1.15.14->boto3)
You are using pip version 9.0.3, however version 20.1 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
buildingmodernapps:~/environment $

```
import boto3

s3 = boto3.resource('s3')
bucket = s3.Bucket('aws-bma-test')

# the objects are available as a collection on the bucket object
for obj in bucket.objects.all():
    print(obj.key, obj.last_modified)

# access the client from the resource
s3_client = boto3.resource('s3').meta.client
```

calling boto3.resource, passing in the service name S3, that's going to give me the resource interface to AWS.

```
import boto3

s3 = boto3.resource('s3')
bucket = s3.Bucket('aws-bma-test')

# the objects are available as a collection on the bucket object
for obj in bucket.objects.all():
    print(obj.key, obj.last_modified)

# access the client from the resource
s3_client = boto3.resource('s3').meta.client
```

Then if I want to get the bucket from S3, I call S3.Bucket.
So that gives me back the bucket as an *object* instead of as a Python dictionary.

```
import boto3

s3 = boto3.resource('s3')
bucket = s3.Bucket('aws-bma-test')

# the objects are available as a collection on the bucket object
for obj in bucket.objects.all():
    print(obj.key, obj.last_modified)

# access the client from the resource
s3_client = boto3.resource('s3').meta.client
```

You can see that's a little bit more readable as the resource and it was with the client.

```
import boto3

s3 = boto3.resource('s3')
bucket = s3.Bucket('aws-bma-test')

# the objects are available as a collection on the bucket object
for obj in bucket.objects.all():
    print(obj.key, obj.last_modified)

# access the client from the resource
s3_client = boto3.resource('s3').meta.client
```

Just to show you how to get the client interface from the resource interface.

Another thing to note about boto3 is that there are something called sessions.

When you create a client or resource a default session is created for you.

The session contains information like your AWS credentials,
and what AWS region you are submitting your API calls to.

Let's talk about the code that
we are going to be using throughout these series of lectures.

The code we're going to go over shortly
is going to read that dragon data, that's a JSON file
that we talked about earlier from an S3 bucket, using S3 Select.

S3 Select is of feature of S3 that allows you to read
a subset of data out of an S3 object using SQL queries.

Being able to query a subset of your data
from an S3 object is actually really powerful.

S3 charges you based on the number of requests, the amount of data
and the data transferred outside of the region.

Note:

S3 Select only supports certain formats of data to be queried.

And in our situation the data is in JSON format, which is supported by S3 Select.

Another AWS service we will be using in our code is AWS Systems Manager Parameter Store.

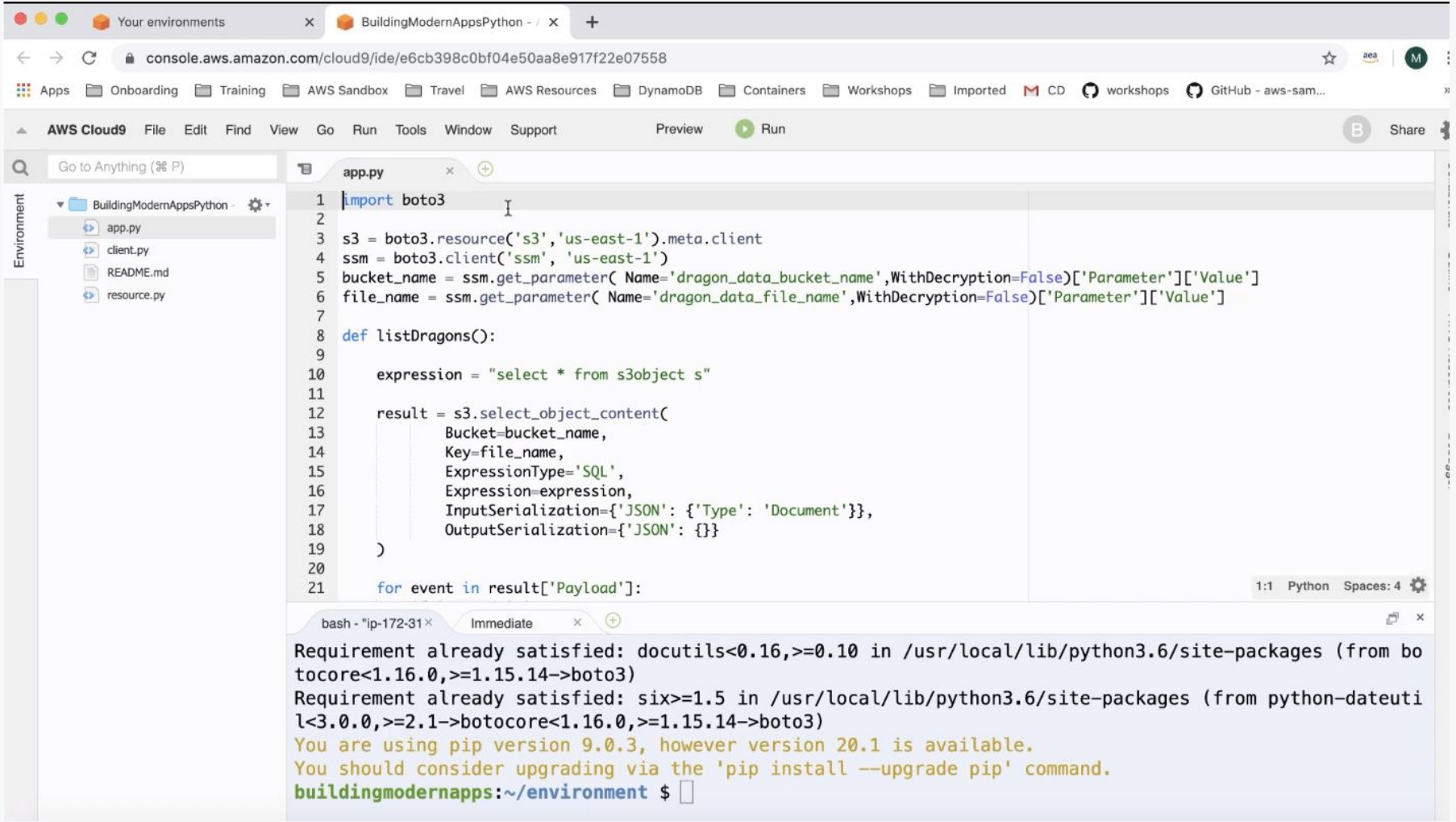
Parameter Store lets you store and retrieve key value pairs that are user defined.

And we're going to use this service to store
our S3 bucket name where the dragon data is located as well as the file name.

That way if the bucket or file ever changes,
we won't need to go back in and change the code.

Instead, the parameter can be updated in Parameter Store
and the code will always pull the latest information every time it is run.

Now go over the file that contains the code that we will be using throughout the course.


Your environments
BuildingModernAppsPython -
console.aws.amazon.com/cloud9/ide/e6cb398c0bf04e50aa8e917f22e07558
Apps
Onboarding
Training
AWS Sandbox
Travel
AWS Resources
DynamoDB
Containers
Workshops
Imported
CD
workshops
GitHub - aws-sam...
AWS Cloud9
File
Edit
Find
View
Go
Run
Tools
Window
Support
Preview
Run
Share
Go to Anything (⌘ P)
Environment
BuildingModernAppsPython
app.py
client.py
README.md
resource.py
app.py
import boto3
s3 = boto3.resource('s3','us-east-1').meta.client
ssm = boto3.client('ssm', 'us-east-1')
bucket_name = ssm.get_parameter(Name='dragon_data_bucket_name',WithDecryption=False)['Parameter']['Value']
file_name = ssm.get_parameter(Name='dragon_data_file_name',WithDecryption=False)['Parameter']['Value']
def listDragons():
expression = "select * from s3object s"
result = s3.select_object_content(
Bucket=bucket_name,
Key=file_name,
ExpressionType='SQL',
Expression=expression,
InputSerialization={'JSON': {'Type': 'Document'}},
OutputSerialization={'JSON': {}}
)
for event in result['Payload']:
1:1
Python
Spaces: 4
bash - "ip-172-31
Immediate
Requirement already satisfied: docutils<0.16,>=0.10 in /usr/local/lib/python3.6/site-packages (from botocore<1.16.0,>=1.15.14->boto3)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.6/site-packages (from python-dateutil<3.0.0,>=2.1->botocore<1.16.0,>=1.15.14->boto3)
You are using pip version 9.0.3, however version 20.1 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
buildingmodernapps:~/environment $

```
import boto3

s3 = boto3.resource('s3','us-east-1').meta.client
ssm = boto3.client('ssm', 'us-east-1')
bucket_name = ssm.get_parameter( Name='dragon_data_bucket_name',WithDecryption=False)['Parameter']['Value']
file_name = ssm.get_parameter( Name='dragon_data_file_name',WithDecryption=False)['Parameter']['Value']

def listDragons():

    expression = "select * from s3object s"

    result = s3.select_object_content(
            Bucket=bucket_name,
            Key=file_name,
            ExpressionType='SQL',
            Expression=expression,
            InputSerialization={'JSON': {'Type': 'Document'}},
            OutputSerialization={'JSON': {}}
    )

    for event in result['Payload']:
        if 'Records' in event:
            print(event['Records']['Payload'].decode('utf-8'))

listDragons()
```

Creating the S3 client by creating the resource in this specific region, us-east-1, and then call .meta.client, which is allowing to access the client through the resource.

```
import boto3

s3 = boto3.resource('s3','us-east-1').meta.client
ssm = boto3.client('ssm', 'us-east-1')
bucket_name = ssm.get_parameter( Name='dragon_data_bucket_name',WithDecryption=False)['Parameter']['Value']
file_name = ssm.get_parameter( Name='dragon_data_file_name',WithDecryption=False)['Parameter']['Value']

def listDragons():

    expression = "select * from s3object s"

    result = s3.select_object_content(
            Bucket=bucket_name,
            Key=file_name,
            ExpressionType='SQL',
            Expression=expression,
            InputSerialization={'JSON': {'Type': 'Document'}},
            OutputSerialization={'JSON': {}}
    )

    for event in result['Payload']:
        if 'Records' in event:
            print(event['Records']['Payload'].decode('utf-8'))

listDragons()
```

Then create the client for Service Systems Manager (SSM here).
SSM being the service name, and Parameter Store is a part of the service Systems Manager.

```
import boto3

s3 = boto3.resource('s3','us-east-1').meta.client
ssm = boto3.client('ssm', 'us-east-1')
bucket_name = ssm.get_parameter( Name='dragon_data_bucket_name',WithDecryption=False)['Parameter']['Value']
file_name = ssm.get_parameter( Name='dragon_data_file_name',WithDecryption=False)['Parameter']['Value']

def listDragons():

    expression = "select * from s3object s"

    result = s3.select_object_content(
            Bucket=bucket_name,
            Key=file_name,
            ExpressionType='SQL',
            Expression=expression,
            InputSerialization={'JSON': {'Type': 'Document'}},
            OutputSerialization={'JSON': {}}
    )

    for event in result['Payload']:
        if 'Records' in event:
            print(event['Records']['Payload'].decode('utf-8'))

listDragons()
```

Next, we are using Systems Manager to grab the dragon data bucket name and the dragon data file name.
This is grabbing the information from Parameter Store and storing it in the bucket name and file name variables.

```
import boto3

s3 = boto3.resource('s3','us-east-1').meta.client
ssm = boto3.client('ssm', 'us-east-1')
bucket_name = ssm.get_parameter( Name='dragon_data_bucket_name',WithDecryption=False)['Parameter']['Value']
file_name = ssm.get_parameter( Name='dragon_data_file_name',WithDecryption=False)['Parameter']['Value']

def listDragons():

    expression = "select * from s3object s"

    result = s3.select_object_content(
            Bucket=bucket_name,
            Key=file_name,
            ExpressionType='SQL',
            Expression=expression,
            InputSerialization={'JSON': {'Type': 'Document'}},
            OutputSerialization={'JSON': {}}
    )

    for event in result['Payload']:
        if 'Records' in event:
            print(event['Records']['Payload'].decode('utf-8'))

listDragons()
```

Define a method called listDragons. Then we are setting up an expression. This is our SQL query that we're going to be using. Right now, it's just select * from s3object s.
Over time, we're going to add to this so that we're actually doing filtering and not just doing select *. But just for ease of getting started, we're just doing select *.

```
import boto3

s3 = boto3.resource('s3','us-east-1').meta.client
ssm = boto3.client('ssm', 'us-east-1')
bucket_name = ssm.get_parameter( Name='dragon_data_bucket_name',WithDecryption=False)['Parameter']['Value']
file_name = ssm.get_parameter( Name='dragon_data_file_name',WithDecryption=False)['Parameter']['Value']

def listDragons():

    expression = "select * from s3object s"

    result = s3.select_object_content(
            Bucket=bucket_name,
            Key=file_name,
            ExpressionType='SQL',
            Expression=expression,
            InputSerialization={'JSON': {'Type': 'Document'}},
            OutputSerialization={'JSON': {}}
    )

    for event in result['Payload']:
        if 'Records' in event:
            print(event['Records']['Payload'].decode('utf-8'))

listDragons()
```

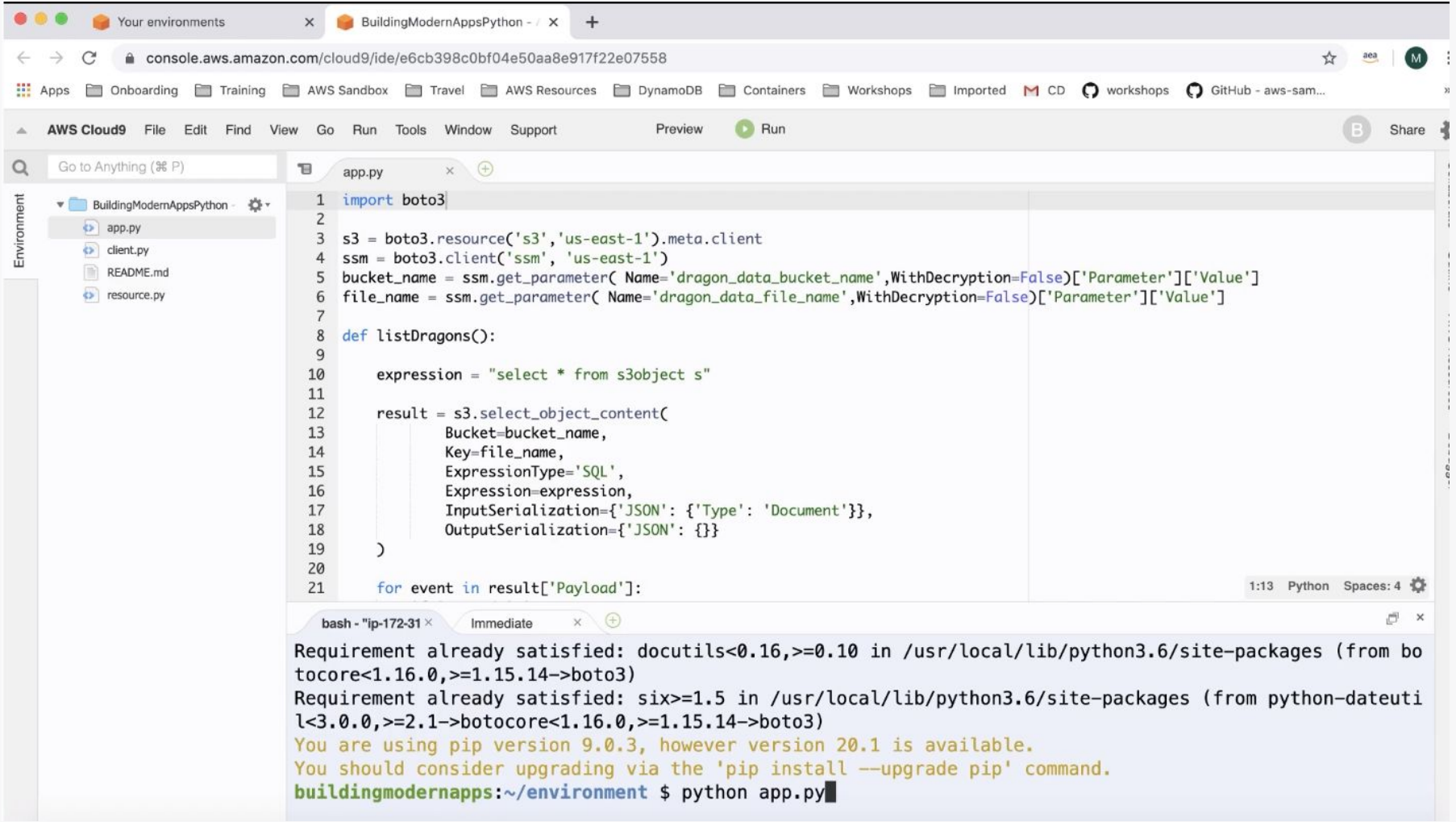
Then we're calling s3.select_object_content. Passing in the bucket name, the file name, the expression type, the expression, the type of the document as well as the type of the output that we want.
So we're pulling the bucket name from up top as well as the file name, the expression type being SQL, the sequel expression being that select * from S3 object. And then we're using JSON from both the input and the output.

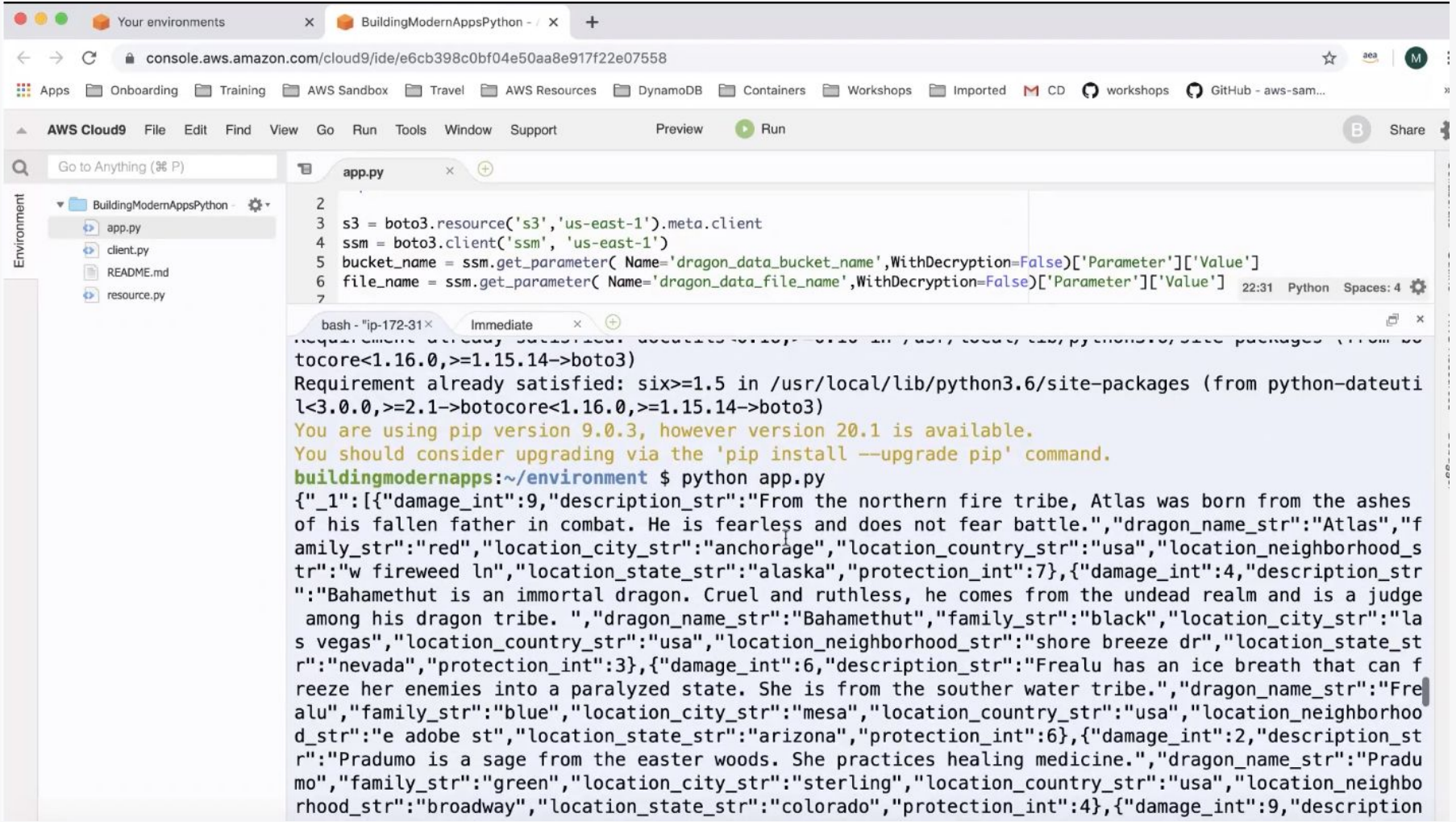
```
import boto3

s3 = boto3.resource('s3','us-east-1').meta.client
ssm = boto3.client('ssm', 'us-east-1')
bucket_name = ssm.get_parameter( Name='dragon_data_bucket_name',WithDecryption=False)['Parameter']['Value']
file_name = ssm.get_parameter( Name='dragon_data_file_name',WithDecryption=False)['Parameter']['Value']

def listDragons():

    expression = "select * from s3object s"

    result = s3.select_object_content(
            Bucket=bucket_name,
            Key=file_name,
            ExpressionType='SQL',
            Expression=expression,
            InputSerialization={'JSON': {'Type': 'Document'}},
            OutputSerialization={'JSON': {}}
    )

    for event in result['Payload']:
        if 'Records' in event:
            print(event['Records']['Payload'].decode('utf-8'))

listDragons()
```

We are looping over all of the events in the result that came back from that API call.
And then we are just printing out those records.

```
import boto3

s3 = boto3.resource('s3','us-east-1').meta.client
ssm = boto3.client('ssm', 'us-east-1')
bucket_name = ssm.get_parameter( Name='dragon_data_bucket_name',WithDecryption=False)['Parameter']['Value']
file_name = ssm.get_parameter( Name='dragon_data_file_name',WithDecryption=False)['Parameter']['Value']

def listDragons():

    expression = "select * from s3object s"

    result = s3.select_object_content(
            Bucket=bucket_name,
            Key=file_name,
            ExpressionType='SQL',
            Expression=expression,
            InputSerialization={'JSON': {'Type': 'Document'}},
            OutputSerialization={'JSON': {}}
    )

    for event in result['Payload']:
        if 'Records' in event:
            print(event['Records']['Payload'].decode('utf-8'))

listDragons()
```

Call listDragons, which call this method and actually kick it off when I run this Python file.


Your environments
BuildingModernAppsPython -
console.aws.amazon.com/cloud9/ide/e6cb398c0bf04e50aa8e917f22e07558
Apps
Onboarding
Training
AWS Sandbox
Travel
AWS Resources
DynamoDB
Containers
Workshops
Imported
CD
workshops
GitHub - aws-sam...
AWS Cloud9
File
Edit
Find
View
Go
Run
Tools
Window
Support
Preview
Run
Share
Go to Anything (⌘ P)
Environment
BuildingModernAppsPython
app.py
client.py
README.md
resource.py
app.py
import boto3
s3 = boto3.resource('s3','us-east-1').meta.client
ssm = boto3.client('ssm', 'us-east-1')
bucket_name = ssm.get_parameter(Name='dragon_data_bucket_name',WithDecryption=False)['Parameter']['Value']
file_name = ssm.get_parameter(Name='dragon_data_file_name',WithDecryption=False)['Parameter']['Value']
def listDragons():
expression = "select * from s3object s"
result = s3.select_object_content(
Bucket=bucket_name,
Key=file_name,
ExpressionType='SQL',
Expression=expression,
InputSerialization={'JSON': {'Type': 'Document'}},
OutputSerialization={'JSON': {}}
)
for event in result['Payload']:
1:13
Python
Spaces: 4
bash - "ip-172-31
Immediate
Requirement already satisfied: docutils<0.16,>=0.10 in /usr/local/lib/python3.6/site-packages (from botocore<1.16.0,>=1.15.14->boto3)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.6/site-packages (from python-dateutil<3.0.0,>=2.1->botocore<1.16.0,>=1.15.14->boto3)
You are using pip version 9.0.3, however version 20.1 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
buildingmodernapps:~/environment $ python app.py


Your environments
BuildingModernAppsPython -
console.aws.amazon.com/cloud9/ide/e6cb398c0bf04e50aa8e917f22e07558
Apps
Onboarding
Training
AWS Sandbox
Travel
AWS Resources
DynamoDB
Containers
Workshops
Imported
CD
workshops
GitHub - aws-sam...
AWS Cloud9
File
Edit
Find
View
Go
Run
Tools
Window
Support
Preview
Run
Share
Go to Anything (⌘ P)
Environment
BuildingModernAppsPython
app.py
client.py
README.md
resource.py
app.py
2
3 s3 = boto3.resource('s3','us-east-1').meta.client
4 ssm = boto3.client('ssm', 'us-east-1')
5 bucket_name = ssm.get_parameter(Name='dragon_data_bucket_name',WithDecryption=False)['Parameter']['Value']
6 file_name = ssm.get_parameter(Name='dragon_data_file_name',WithDecryption=False)['Parameter']['Value']
7
22:31
Python
Spaces: 4
bash - "ip-172-31
Immediate
tocore<1.16.0,>=1.15.14->boto3)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.6/site-packages (from python-dateuti
l<3.0.0,>=2.1->botocore<1.16.0,>=1.15.14->boto3)
You are using pip version 9.0.3, however version 20.1 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
buildingmodernapps:~/environment $ python app.py
{"_1":[{"damage_int":9,"description_str":"From the northern fire tribe, Atlas was born from the ashes
of his fallen father in combat. He is fearless and does not fear battle.","dragon_name_str":"Atlas","f
amily_str":"red","location_city_str":"anchorage","location_country_str":"usa","location_neighborhood_s
tr":"w fireweed ln","location_state_str":"alaska","protection_int":7},{"damage_int":4,"description_str
":"Bahamethut is an immortal dragon. Cruel and ruthless, he comes from the undead realm and is a judge
among his dragon tribe. ","dragon_name_str":"Bahamethut","family_str":"black","location_city_str":"la
s vegas","location_country_str":"usa","location_neighborhood_str":"shore breeze dr","location_state_st
r":"nevada","protection_int":3},{"damage_int":6,"description_str":"Frealu has an ice breath that can f
reeze her enemies into a paralyzed state. She is from the souther water tribe.","dragon_name_str":"Fre
alu","family_str":"blue","location_city_str":"mesa","location_country_str":"usa","location_neighborhoo
d_str":"e adobe st","location_state_str":"arizona","protection_int":6},{"damage_int":2,"description_st
r":"Pradumo is a sage from the easter woods. She practices healing medicine.","dragon_name_str":"Pradu
mo","family_str":"green","location_city_str":"sterling","location_country_str":"usa","location_neighbo
rhood_str":"broadway","location_state_str":"colorado","protection_int":4},{"damage_int":9,"description

Note: Nowhere in my code did I specify any AWS credentials!

While you could pass your credentials into the client,
I would not recommend that you do that.

Because then anyone who has your code also has your credentials.

The AWS SDK will handle locating the credentials for you.

The SDK will check for your credentials in this order:

- First, it will see if you pass the credentials as parameters in the boto client method.
  Don't do this unless you absolutely have to.
- Then it will check the session to see if you pass the credentials and its parameters.
  Again, don't do this.
- Next, it will check the environment variables. Then it will check the credentials in the config files that get created when you run the AWS configure command from the command line.
  This is not that much better than including the credentials in your code.
- And finally, it will check the instance metadata service
  which exists on Amazon EC2 instances that have an IAM role associated.
  That is the preferred way to use AWS credentials.

The IAM role will allow your code to access temporary credentials,
which is more secure than embedding them or hard coding them or using a file.

Since we are developing on a Cloud9 instance,
the credentials will be retrieved from the managed credentials that we talked about earlier.

If you were developing locally, you would want to make sure that
your configuration and credentials files were configured with your
region, AWS access key ID and AWS secret access key
since your local machine doesn't have an instance metadata service.

Using Temporary Credentials in AWS Cloud9

Let’s explain how AWS manage temporary credentials in Cloud9.

You may think that Cloud9 is just like an IDE in a bash terminal on your laptop,
which is almost right.

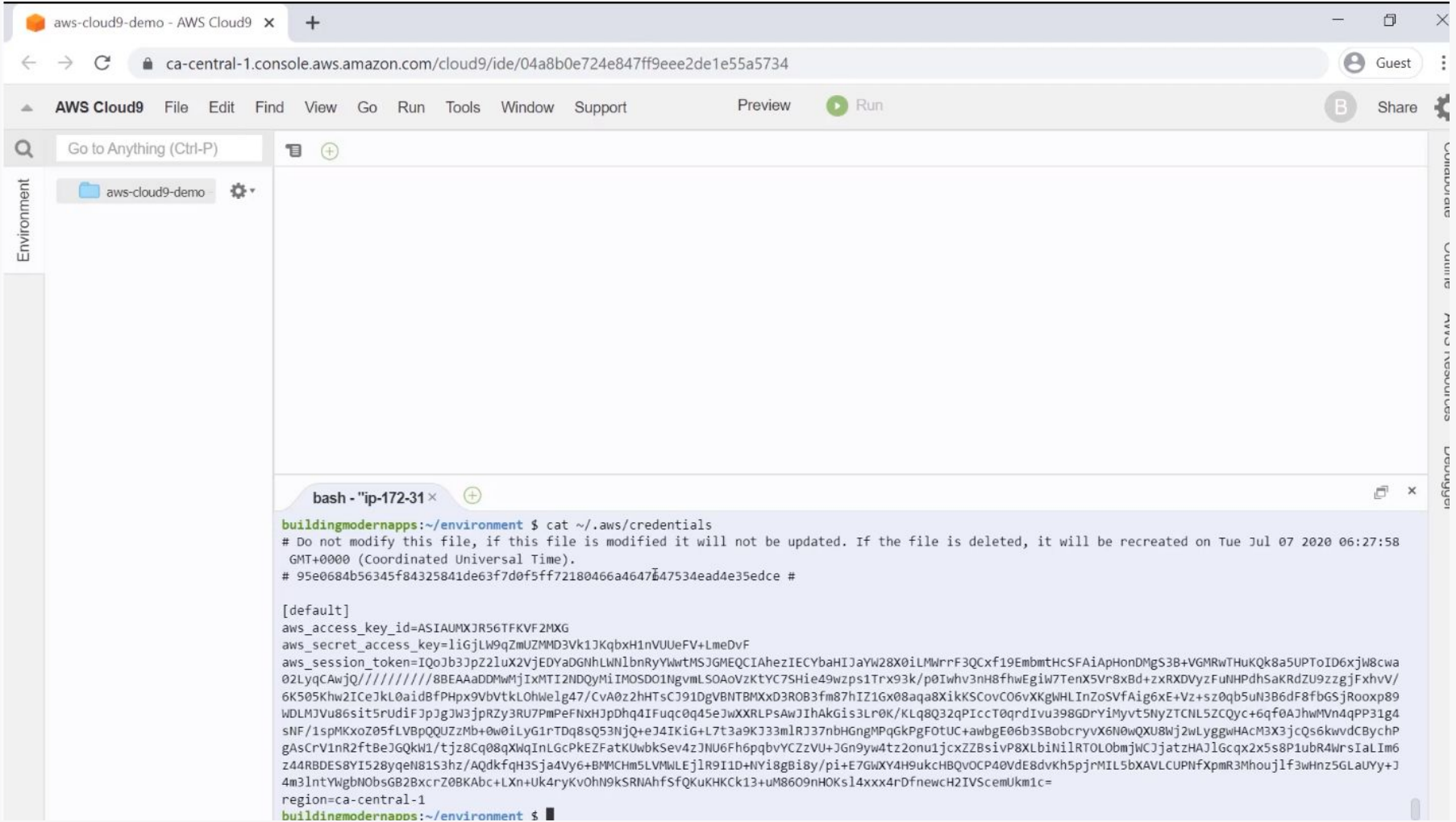
But there's one major difference by default around credentials.

Cloud9 automatically creates managed temporary credentials,
when it's hosted on an EC2 instance.

Credentials can be stored in your home directory,
under the .aws folder in the file "credentials".

run this in your cloud9 to see your temporary credentials:

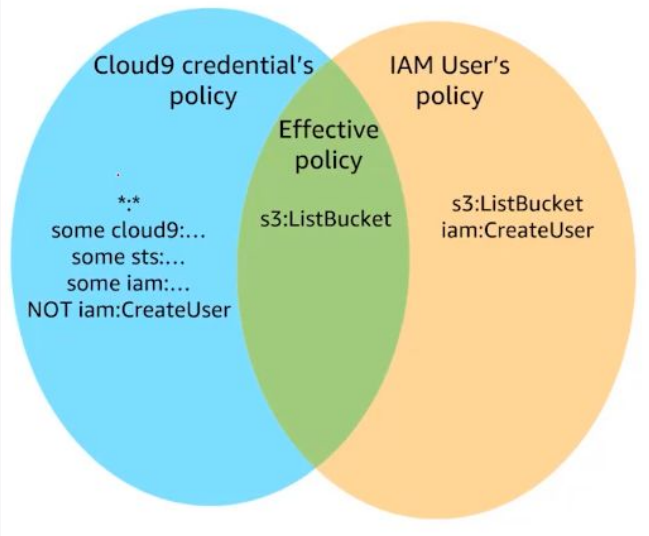
```
$ cat ~/.aws/credentials
```


aws-cloud9-demo - AWS Cloud9
ca-central-1.console.aws.amazon.com/cloud9/ide/04a8b0e724e847ff9eee2de1e55a5734
Guest
AWS Cloud9 File Edit Find View Go Run Tools Window Support
Preview
Run
Share
Go to Anything (Ctrl-P)
Environment
aws-cloud9-demo
bash - "ip-172-31
buildingmodernapps:~/environment $ cat ~/.aws/credentials
Do not modify this file, if this file is modified it will not be updated. If the file is deleted, it will be recreated on Tue Jul 07 2020 06:27:58 GMT+0000 (Coordinated Universal Time).
95e0684b56345f84325841de63f7d0f5ff72180466a4647b47534ead4e35edce
[default]
aws_access_key_id=ASIAUMXJR56TFKVF2MXG
aws_secret_access_key=liGjLW9qZmUZMMD3Vk1JKqbxH1nVUUeFV+LmeDvF
aws_session_token=IQoJb3JpZ2luX2VjEDYaDGNhLWNlbnRyYWwtMSJGMEQCIAhezIECYbaHIJaYW28X0iLMWrrF3QCxf19EmbmtHcSFAiApHonDMgS3B+VGMRwTHuKQk8a5UPToID6xjW8cwa02LyqCAwjQ/////////8BEAAaDDMwMjIxMTI2NDQyMiIMOSDO1NgvmLSOAoVzKtYC7SHie49wzps1Trx93k/p0Iwhv3nH8fhwEgiW7TenX5Vr8xBd+zxRXDVyzFuNHPdhSaKRdZU9zzgjFxhvV/6K505Khw2ICeJkL0aidBfPHpx9VbVtkLOhWelg47/CvA0z2hHTsCJ91DgVBNTBMXxD3ROB3fm87hIZ1Gx08aqa8XikKSCovCO6vXKgWHLInZoSVfAig6xE+Vz+sz0qb5uN3B6dF8fbGSjRooxp89WDLMJVu86sit5rUdiFJpJgJW3jpRZy3RU7PmPeFNxHJpDhq4IFuqc0q45eJwXXRLPsAwJIhAkGis3Lr0K/KLq8Q32qPIccT0qrdIvu398GDrYiMyvt5NyZTCNL5ZCQyc+6qf0AJhwMVn4qPP31g4sNF/1spMKxoZ05fLVBpQQUZzMb+0w0iLyG1rTDq8sQ53NjQ+eJ4IKiG+L7t3a9KJ33mlRJ37nbHGngMPqGkPgFOtUC+awbgE06b3SBobcryvX6N0wQXU8Wj2wLyggwHAcM3X3jcQs6kwvdCBychPgAsCrV1nR2ftBeJGQkW1/tjz8Cq08qXWqInLGcPkEZFatKUwbkSev4zJNU6Fh6pqbvYCZzVU+JGn9yw4tz2onu1jcxZZBsivP8XLbiNilRTOLObmjWCjjatzHAJlGcqx2x5s8P1ubR4WrsIaLIm6z44RBDES8YI528yqeN81S3hz/AQdkfqH3Sja4Vy6+BMMCHm5LVMWLEjlR9I1D+NYi8gBi8y/pi+E7GWXY4H9ukcHBQvOCP40VdE8dvKh5pjrMIL5bXAVLCUPNfXpmR3MhoujIf3wHnz5GLaUYy+J4m3lntYWgbNObsGB2BxcrZ0BKAbc+LXn+Uk4ryKvOhN9kSRNAhfSfQKuKHKCk13+uM86O9nHOKsl4xxx4rDfnewcH2IVScemUkm1c=
region=ca-central-1
buildingmodernapps:~/environment $

These credentials, they're no longer valid at the time this lecture because every five minutes Cloud9 will automatically rewrite this file with brand new credentials.

Note: Those managed temporary credentials allow all actions for all AWS resources, with some very important exceptions for the Cloud9, SDS and IAM service where it only allows for some actions.

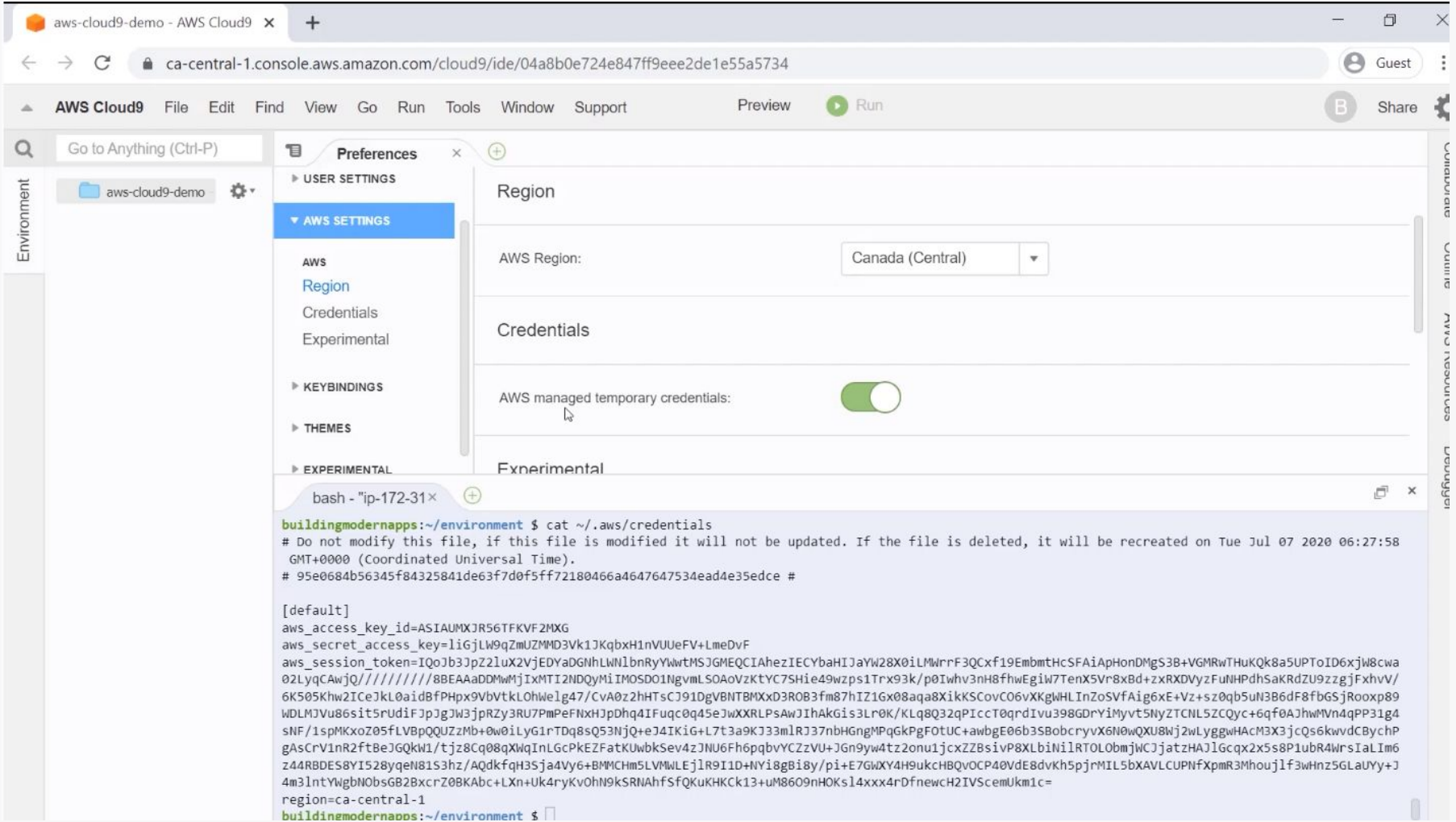
For example, the `iam:CreateUser` action isn't allowed.

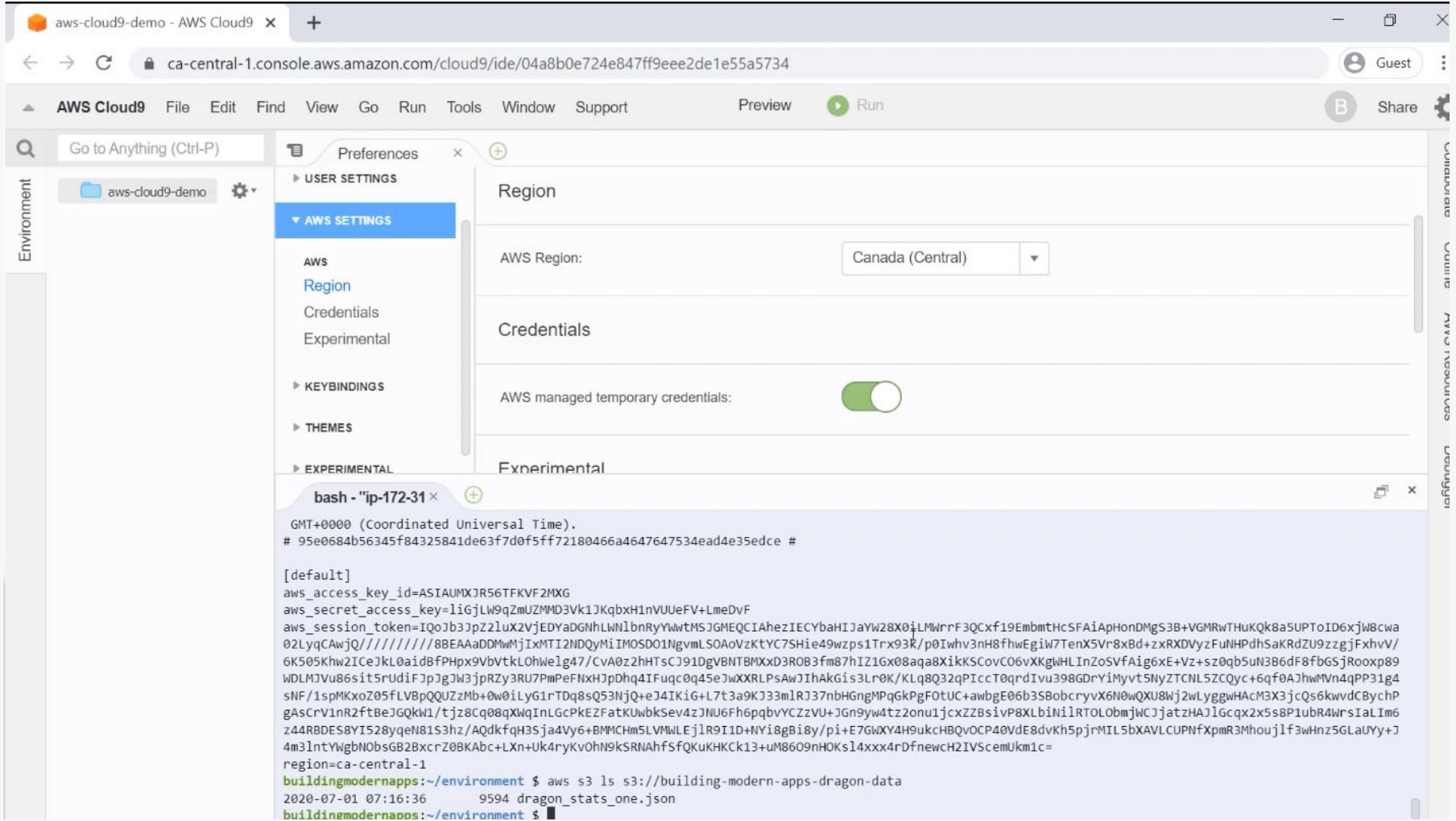

Cloud9 credential's policy
:
some cloud9:...
some sts:...
some iam:...
NOT iam:CreateUser
Effective policy
s3:ListBucket
IAM User's policy
s3:ListBucket
iam:CreateUser

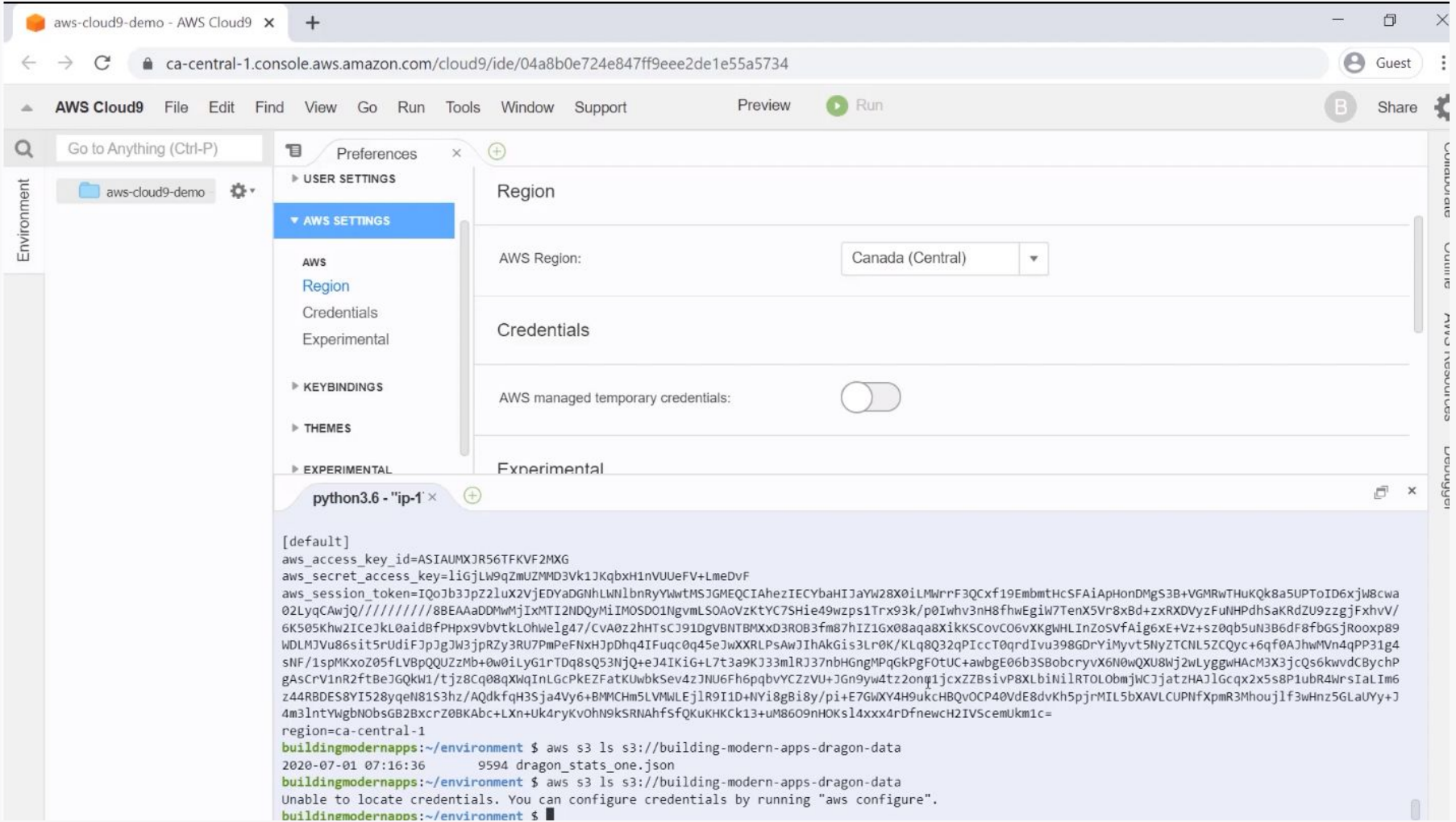
What if you **don't** want to use these temporary credentials,
either because you want to share your environment with someone else,
and you don't want them to have the same access as you,
or because you are trying to do certain actions that
the managed temporary credentials won't allow you to do, like creating an IAM user?

The first thing to do is to go and disable that feature.

Clicking on the AWS Cloud9 menu, select Preferences, and then click on AWS settings, and then under Credentials, uncheck the toggle for AWS managed temporary credentials.


aws-cloud9-demo - AWS Cloud9
ca-central-1.console.aws.amazon.com/cloud9/ide/04a8b0e724e847ff9eee2de1e55a5734
Guest
AWS Cloud9 File Edit Find View Go Run Tools Window Support
Preview
Run
Share
Go to Anything (Ctrl-P)
Environment
aws-cloud9-demo
Preferences
USER SETTINGS
AWS SETTINGS
AWS
Region
Credentials
Experimental
KEYBINDINGS
THEMES
EXPERIMENTAL
Region
AWS Region:
Canada (Central)
Credentials
AWS managed temporary credentials:
Experimental
bash - "ip-172-31
buildingmodernapps:~/environment $ cat ~/.aws/credentials
Do not modify this file, if this file is modified it will not be updated. If the file is deleted, it will be recreated on Tue Jul 07 2020 06:27:58 GMT+0000 (Coordinated Universal Time).
95e0684b56345f84325841de63f7d0f5ff72180466a4647647534ead4e35edce
[default]
aws_access_key_id=ASIAUMXJR56TFKVF2MXG
aws_secret_access_key=liGjLW9qZmUZMMD3Vk1JKqbxH1nVUUeFV+LmeDvF
aws_session_token=IQoJb3JpZ2luX2VjEDYaDGNhLWNlbnRyYWwtMSJGMEQCIAhezIECYbaHIJaYW28X0iLMWrrF3QCxf19EmbmtHcSFAiApHonDMgS3B+VGMRwTHuKQk8a5UPToID6xjW8cwa02LyqCAwjQ/////////8BEAAaDDMwMjIxMTI2NDQyMiIMOSDO1NgvmLSOAoVzKtYC7SHie49wzps1Trx93k/p0Iwhv3nH8fhwEgiW7TenX5Vr8xBd+zxRXDVyzFuNHPdhSaKRdZU9zzgjFxhvV/6K505Khw2ICeJkL0aidBfPHpx9VbVtkLOhWelg47/CvA0z2hHTsCJ91DgVBNTBMXxD3ROB3fm87hIZ1Gx08aqa8XikKSCovCO6vXKgWHLInZoSVfAig6xE+Vz+sz0qb5uN3B6dF8fbGSjRooxp89WDLMJVu86sit5rUdiFJpJgJW3jpRZy3RU7PmPeFNxHJpDhq4IFuqc0q45eJwXXRLPsAwJIhAkGis3Lr0K/KLq8Q32qPIccT0qrdIvu398GDrYiMyvt5NyZTCNL5ZCQyc+6qf0AJhwMVn4qPP31g4sNF/1spMKxoZ05fLVBpQQUZzMb+0w0iLyG1rTDq8sQ53NjQ+eJ4IKiG+L7t3a9KJ33mlRJ37nbHGngMPqGkPgFOtUC+awbgE06b3SBobcryvX6N0wQXU8Wj2wLyggwHAcM3X3jcQs6kwvdCBychPgAsCrV1nR2ftBeJGQkW1/tjz8Cq08qXWqInLGcPkEZFatKUwbkSev4zJNU6Fh6pqbvYCZzVU+JGn9yw4tz2onu1jcxZZBsivP8XLbiNilRTOLObmjWCjjatzHAJlGcqx2x5s8P1ubR4WrsIaLIm6z44RBDES8YI528yqeN81S3hz/AQdkfqH3Sja4Vy6+BMMCHm5LVMWLEjlR9I1D+NYi8gBi8y/pi+E7GWXY4H9ukcHBQvOCP40VdE8dvKh5pjrMIL5bXAVLCUPNfXpmR3MhoujIf3wHnz5GLaUYy+J4m3lntYWgbNObsGB2BxcrZ0BKAbc+LXn+Uk4ryKvOhN9kSRNAhfSfQKuKHKCk13+uM86O9nHOKsl4xxx4rDfnewcH2IVScemUkm1c=
region=ca-central-1
buildingmodernapps:~/environment $


aws-cloud9-demo - AWS Cloud9
ca-central-1.console.aws.amazon.com/cloud9/ide/04a8b0e724e847ff9eee2de1e55a5734
Guest
AWS Cloud9 File Edit Find View Go Run Tools Window Support
Preview
Run
Share
Go to Anything (Ctrl-P)
Environment
aws-cloud9-demo
Preferences
USER SETTINGS
AWS SETTINGS
AWS
Region
Credentials
Experimental
KEYBINDINGS
THEMES
EXPERIMENTAL
Region
AWS Region:
Canada (Central)
Credentials
AWS managed temporary credentials:
Experimental
bash - "ip-172-31
GMT+0000 (Coordinated Universal Time).
95e0684b56345f84325841de63f7d0f5ff72180466a4647647534ead4e35edce
[default]
aws_access_key_id=ASIAUMXJR56TFKVF2MXG
aws_secret_access_key=liGjLW9qZmUZMMD3Vk1JKqbxH1nVUUeFV+LmeDvF
aws_session_token=IQoJb3JpZ2luX2VjEDYaDGNhLWNlbnRyYWwtMSJGMEQCIAhezIECYbaHIJaYW28X0iLMWrrF3QCxf19EmbmtHcSFAiApHonDMgS3B+VGMRwTHuKQk8a5UPToID6xjW8cwa
02LyqCAwjQ/////////8BEAAaDDMwMjIxMTI2NDQyMiIMOSDO1NgvmLSOAoVzKtYC7SHie49wzps1Trx93k/p0Iwhv3nH8fhwEgiW7TenX5Vr8xBd+zxRXDVyzFuNHPdhSaKRdZU9zzgjFxhvV/
6K505Khw2ICeJkL0aidBfPHpx9VbVtkLOhWelg47/CvA0z2hHTsCJ91DgVBNTBMXxD3ROB3fm87hIZ1Gx08aqa8XikKSCovCO6vXKgWHLInZoSVfAig6xE+Vz+sz0qb5uN3B6dF8fbGSjRooxp89
WDLMJVu86sit5rUdiFJpJgJW3jpRZy3RU7PmPeFNxHJpDhq4IFuqc0q45eJwXXRLPsAwJIhAkGis3Lr0K/KLq8Q32qPIccT0qrdIvu398GDrYiMyvt5NyZTCNL5ZCQyc+6qf0AJhwMVn4qPP31g4
sNF/1spMKxoZ05fLVBpQQUZzMb+0w0iLyG1rTDq8sQ53NjQ+eJ4IKiG+L7t3a9KJ33mlRJ37nbHGngMPqGkPgFOtUC+awbgE06b3SBobcryvX6N0wQXU8Wj2wLyggwHAcM3X3jcQs6kwvdCBychP
gAsCrV1nR2ftBeJGQkW1/tjz8Cq08qXWqInLGcPkEZFatKUwbkSev4zJNU6Fh6pqbvYCZzVU+JGn9yw4tz2onu1jcxZZBsivP8XLbiNilRTOLObmjWCjjatzHAJlGcqx2x5s8P1ubR4WrsIaLIm6
z44RBDES8YI528yqeN81S3hz/AQdkfqH3Sja4Vy6+BMMCHm5LVMWLEjlR9I1D+NYi8gBi8y/pi+E7GWXY4H9ukcHBQvOCP40VdE8dvKh5pjrMIL5bXAVLCUPNfXpmR3Mhoujlf3wHnz5GLaUYy+J
4m3lntYWgbNObsGB2BxcrZ0BKAbc+LXn+Uk4ryKvOhN9kSRNAhfSfQKuKHKCk13+uM86O9nHOKsl4xxx4rDfnewcH2IVScemUkm1c=
region=ca-central-1
buildingmodernapps:~/environment $ aws s3 ls s3://building-modern-apps-dragon-data
2020-07-01 07:16:36 9594 dragon_stats_one.json
buildingmodernapps:~/environment $


aws-cloud9-demo - AWS Cloud9
ca-central-1.console.aws.amazon.com/cloud9/ide/04a8b0e724e847ff9eee2de1e55a5734
Guest
AWS Cloud9 File Edit Find View Go Run Tools Window Support
Preview
Run
Share
Go to Anything (Ctrl-P)
Environment
aws-cloud9-demo
Preferences
USER SETTINGS
AWS SETTINGS
AWS
Region
Credentials
Experimental
KEYBINDINGS
THEMES
EXPERIMENTAL
Region
AWS Region:
Canada (Central)
Credentials
AWS managed temporary credentials:
Experimental
python3.6 - "ip-1
[default]
aws_access_key_id=ASIAUMXJR56TFKVF2MXG
aws_secret_access_key=liGjLW9qZmUZMMD3Vk1JKqbxH1nVUUeFV+LmeDvF
aws_session_token=IQoJb3JpZ2luX2VjEDYaDGNhLWNlbnRyYWwtMSJGMEQCIAhezIECYbaHIJaYW28X0iLMWrrF3QCxf19EmbmtHcSFAiApHonDMgS3B+VGMRwTHuKQk8a5UPToID6xjW8cwa
02LyqCAwjQ/////////8BEAAaDDMwMjIxMTI2NDQyMiIMOSDO1NgvmLSOAoVzKtYC7SHie49wzps1Trx93k/p0Iwhv3nH8fhwEgiW7TenX5Vr8xBd+zxRXDVyzFuNHPdhSaKRdZU9zzgjFxhvV/
6K505Khw2ICeJkL0aidBfPHpx9VbVtkLOhWelg47/CvA0z2hHTsCJ91DgVBNTBMXxD3ROB3fm87hIZ1Gx08aqa8XikKSCovCO6vXKgWHLInZoSVfAig6xE+Vz+sz0qb5uN3B6dF8fbGSjRooxp89
WDLMJVu86sit5rUdiFJpJgJW3jpRZy3RU7PmPeFNxHJpDhq4IFuqc0q45eJwXXRLPsAwJIhAkGis3Lr0K/KLq8Q32qPIccT0qrdIvu398GDrYiMyvt5NyZTCNL5ZCQyc+6qf0AJhwMVn4qPP31g4
sNF/1spMKxoZ05fLVBpQQUZzMb+0w0iLyG1rTDq8sQ53NjQ+eJ4IKiG+L7t3a9KJ33mlRJ37nbHGngMPqGkPgFOtUC+awbgE06b3SBobcryvX6N0wQXU8Wj2wLyggwHAcM3X3jcQs6kwvdCBychP
gAsCrV1nR2ftBeJGQkW1/tjz8Cq08qXWqInLGcPkEZFatKUwbkSev4zJNU6Fh6pqbvYCZzVU+JGn9yw4tz2onq1jcxZZBsivP8XLbiNilRTOLObmjWCJjatzHAJlGcqx2x5s8P1ubR4WrsIaLIm6
z44RBDES8YI528yqeN81S3hz/AQdkfqH3Sja4Vy6+BMMCHm5LVMWLEjlR9I1D+NYi8gBi8y/pi+E7GWXY4H9ukcHBQvOCP40VdE8dvKh5pjrMIL5bXAVLCUPNfXpmR3Mhoujlf3wHnz5GLaUYy+J
4m3lntYWgbNObsGB2BxcrZ0BKAbc+LXn+Uk4ryKvOhN9kSRNAhfSfQKuKHKCk13+uM86O9nHOKsl4xxx4rDfnewcH2IVScemUkm1c=
region=ca-central-1
buildingmodernapps:~/environment $ aws s3 ls s3://building-modern-apps-dragon-data
2020-07-01 07:16:36 9594 dragon_stats_one.json
buildingmodernapps:~/environment $ aws s3 ls s3://building-modern-apps-dragon-data
Unable to locate credentials. You can configure credentials by running "aws configure".
buildingmodernapps:~/environment $

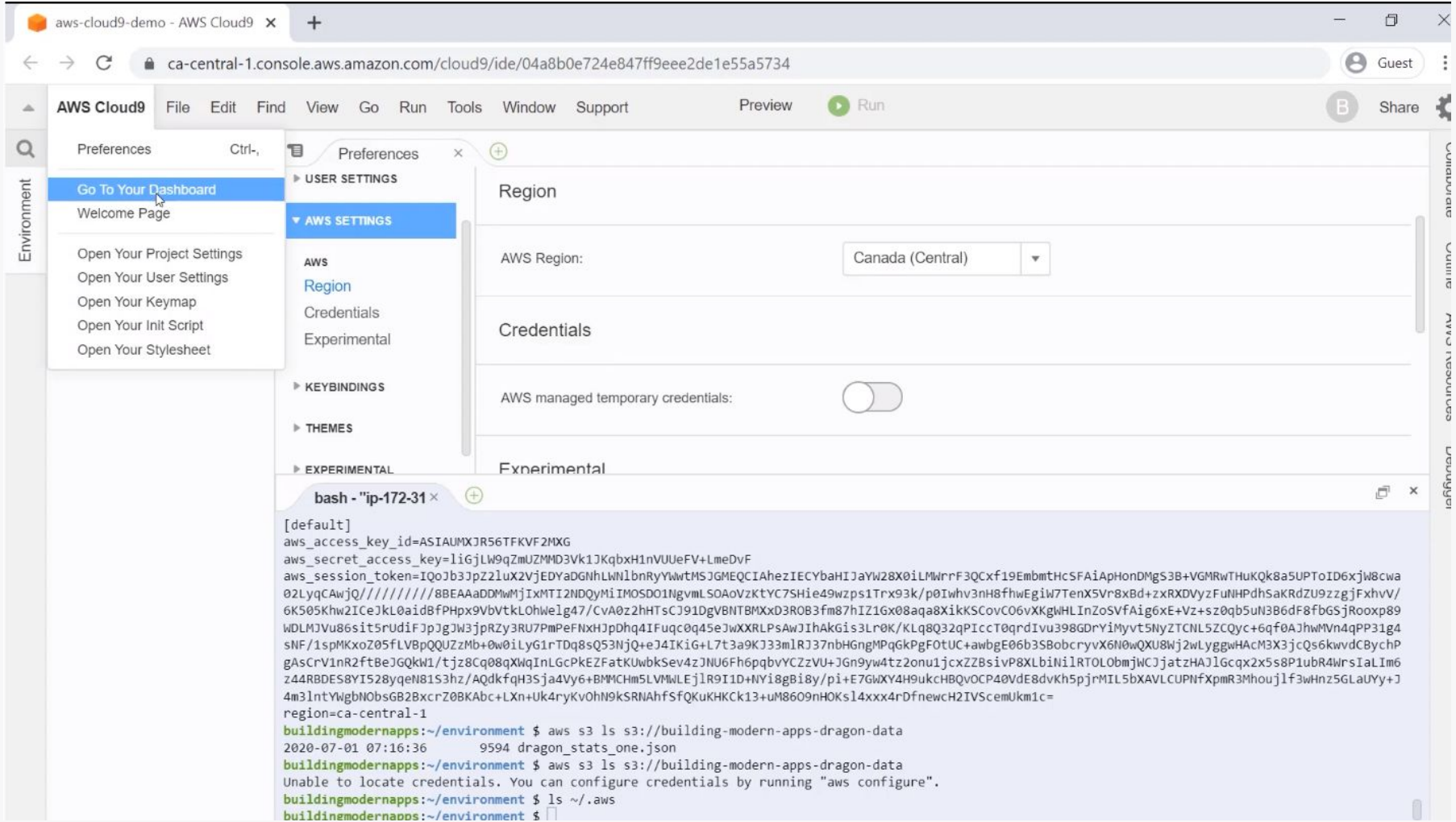
From this point, we have two options to fix this.

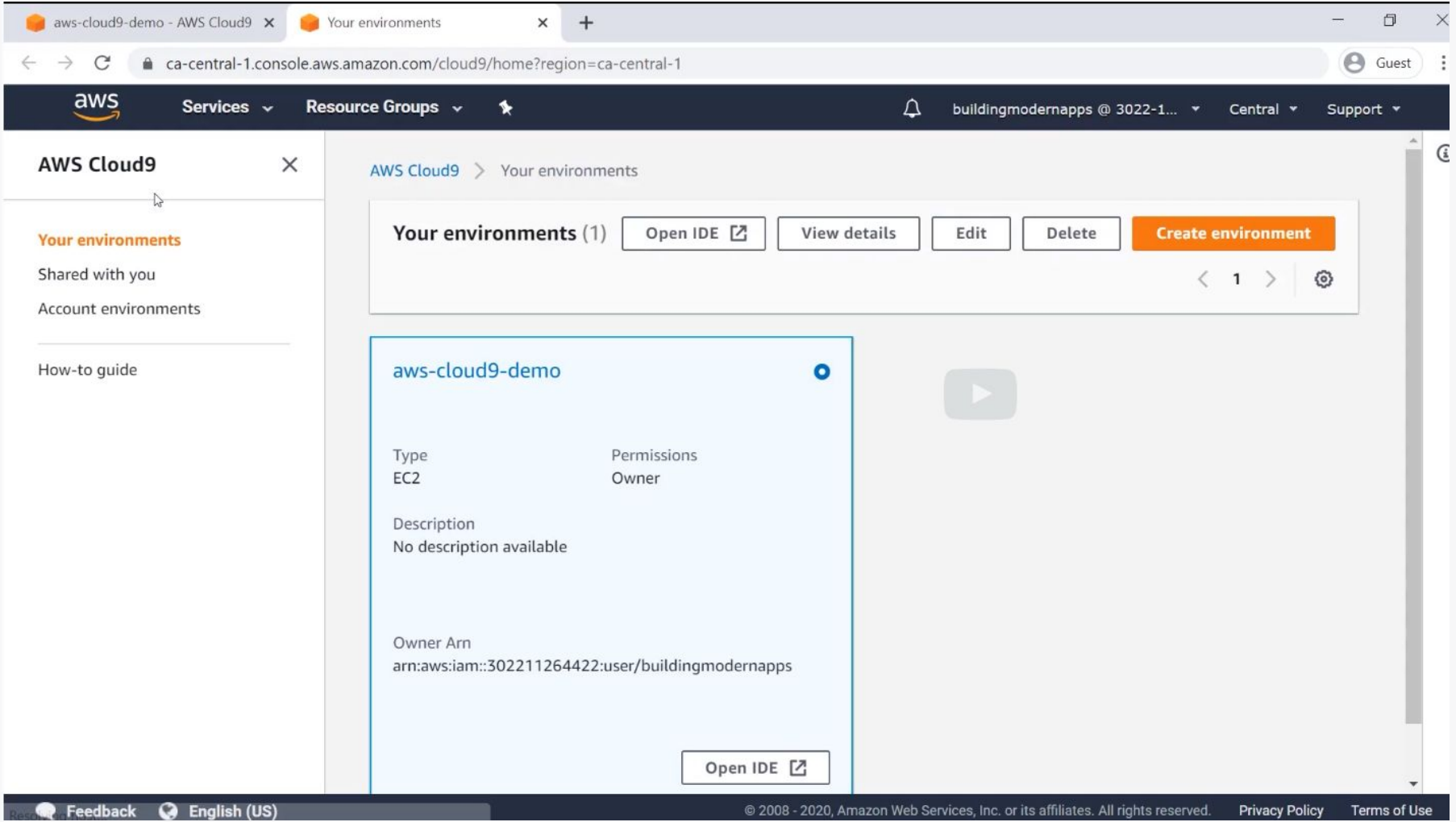
The first option is to attach an IAM role to the EC2 instance that was created by launching this Cloud9 instance.

So let's do that first.

I need to go to the EC2 console to change that role.

To go back to the EC2 console, click on the Cloud9 menu, then Go To Your Dashboard, which brings you to the AWS Console.

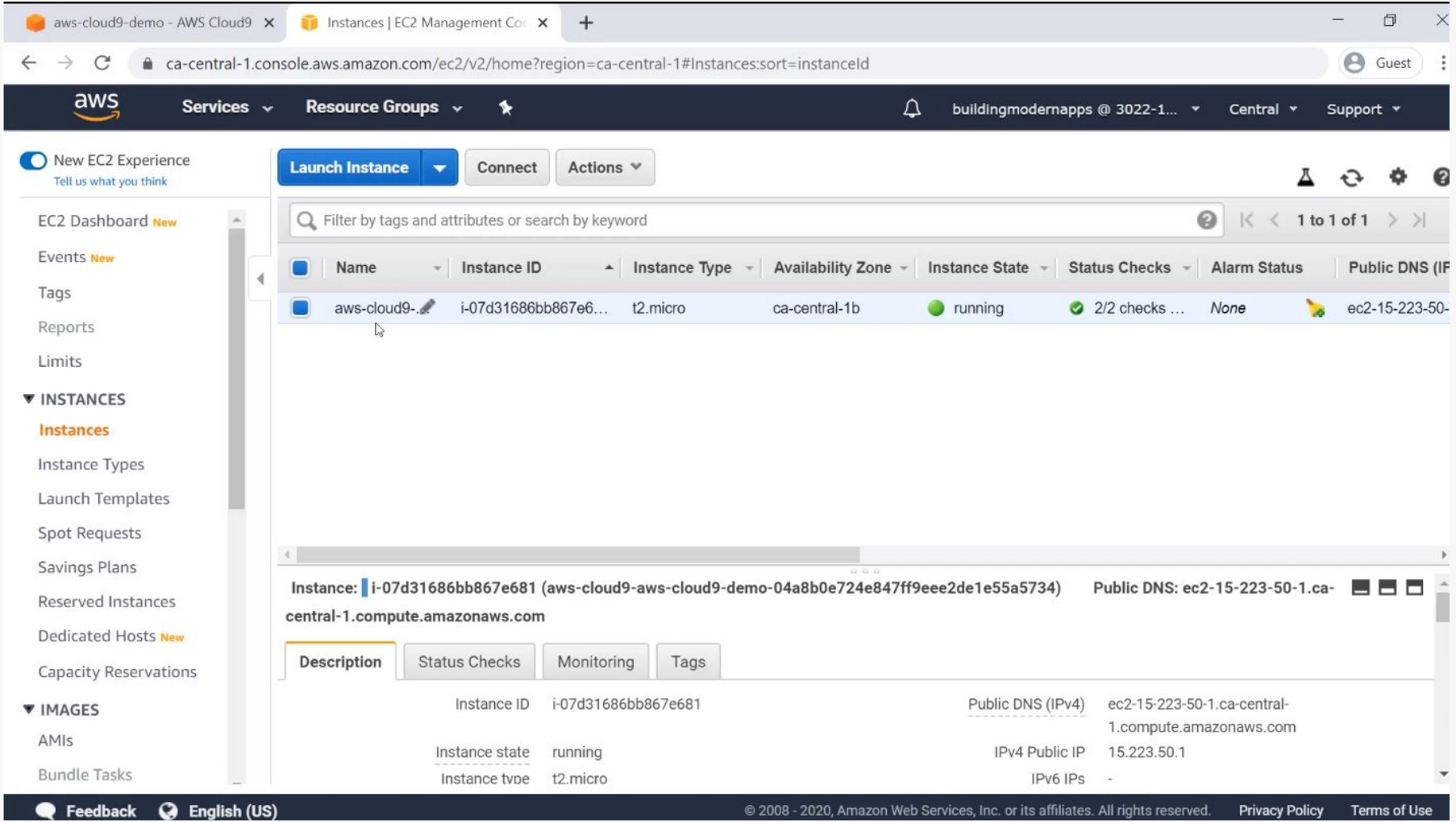

aws-cloud9-demo - AWS Cloud9
ca-central-1.console.aws.amazon.com/cloud9/ide/04a8b0e724e847ff9eee2de1e55a5734
Guest
AWS Cloud9
File
Edit
Find
View
Go
Run
Tools
Window
Support
Preview
Run
Share
Environment
Preferences Ctrl-,
Go To Your Dashboard
Welcome Page
Open Your Project Settings
Open Your User Settings
Open Your Keymap
Open Your Init Script
Open Your Stylesheet
Preferences
USER SETTINGS
AWS SETTINGS
AWS
Region
Credentials
Experimental
KEYBINDINGS
THEMES
EXPERIMENTAL
Region
AWS Region:
Canada (Central)
Credentials
AWS managed temporary credentials:
Experimental
bash - "ip-172-31
[default]
aws_access_key_id=ASIAUMXJR56TFKVF2MXG
aws_secret_access_key=liGjLW9qZmUZMMD3Vk1JKqbxH1nVUUeFV+LmeDvF
aws_session_token=IQoJb3JpZ2luX2VjEDYaDGNhLWNlbnRyYWwtMSJGMEQCIAhezIECYbaHIJaYW28X0iLMWrrF3QCxf19EmbmtHcSFAiApHonDMgS3B+VGMRwTHuKQk8a5UPToID6xjW8cwa02LyqCAwjQ/////////8BEAAaDDMwMjIxMTI2NDQyMiIMOSDO1NgvmLSOAoVzKtYC7SHie49wzps1Trx93k/p0Iwhv3nH8fhwEgiW7TenX5Vr8xBd+zxRXDVyzFuNHPdhSaKRdZU9zzgjFxhvV/6K505Khw2ICeJkL0aidBfPHpx9VbVtkLOhWelg47/CvA0z2hHTsCJ91DgVBNTBMXxD3ROB3fm87hIZ1Gx08aqa8XikKSCovCO6vXKgWHLInZoSVfAig6xE+Vz+sz0qb5uN3B6dF8fbGSjRooxp89WDLMJVu86sit5rUdiFJpJgJW3jpRZy3RU7PmPeFNxHJpDhq4IFuqc0q45eJwXXRLPsAwJIhAkGis3Lr0K/KLq8Q32qPIccT0qrdIvu398GDrYiMyvt5NyZTCNL5ZCQyc+6qf0AJhwMVn4qPP31g4sNF/1spMKxoZ05fLVBpQQUZzMb+0w0iLyG1rTDq8sQ53NjQ+eJ4IKiG+L7t3a9KJ33mlRJ37nbHGngMPqGkPgFOtUC+awbgE06b3SBobcryvX6N0wQXU8Wj2wLyggwHAcM3X3jcQs6kwvdCBychPgAsCrV1nR2ftBeJGQkW1/tjz8Cq08qXWqInLGcPkEZFatKUwbkSev4zJNU6Fh6pqbvYCZzVU+JGn9yw4tz2onu1jcxZZBsivP8XLbiNilRTOLObmjWCJjatzHAJlGcqx2x5s8P1ubR4WrsIaLIm6z44RBDES8YI528yqeN81S3hz/AQdkfqH3Sja4Vy6+BMMCHm5LVMWLEjlR9I1D+NYi8gBi8y/pi+E7GWXY4H9ukcHBQvOCP40VdE8dvKh5pjrMIL5bXAVLCUPNfXpmR3MhoujlF3wHnz5GLaUYy+J4m3lntYWgbNObsGB2BxcrZ0BKAbc+LXn+Uk4ryKvOhN9kSRNAhfSfQKuKHKCk13+uM86O9nHOKsl4xxx4rDfnewcH2IVScemUkm1c=
region=ca-central-1
buildingmodernapps:~/environment $ aws s3 ls s3://building-modern-apps-dragon-data
2020-07-01 07:16:36 9594 dragon_stats_one.json
buildingmodernapps:~/environment $ aws s3 ls s3://building-modern-apps-dragon-data
Unable to locate credentials. You can configure credentials by running "aws configure".
buildingmodernapps:~/environment $ ls ~/.aws

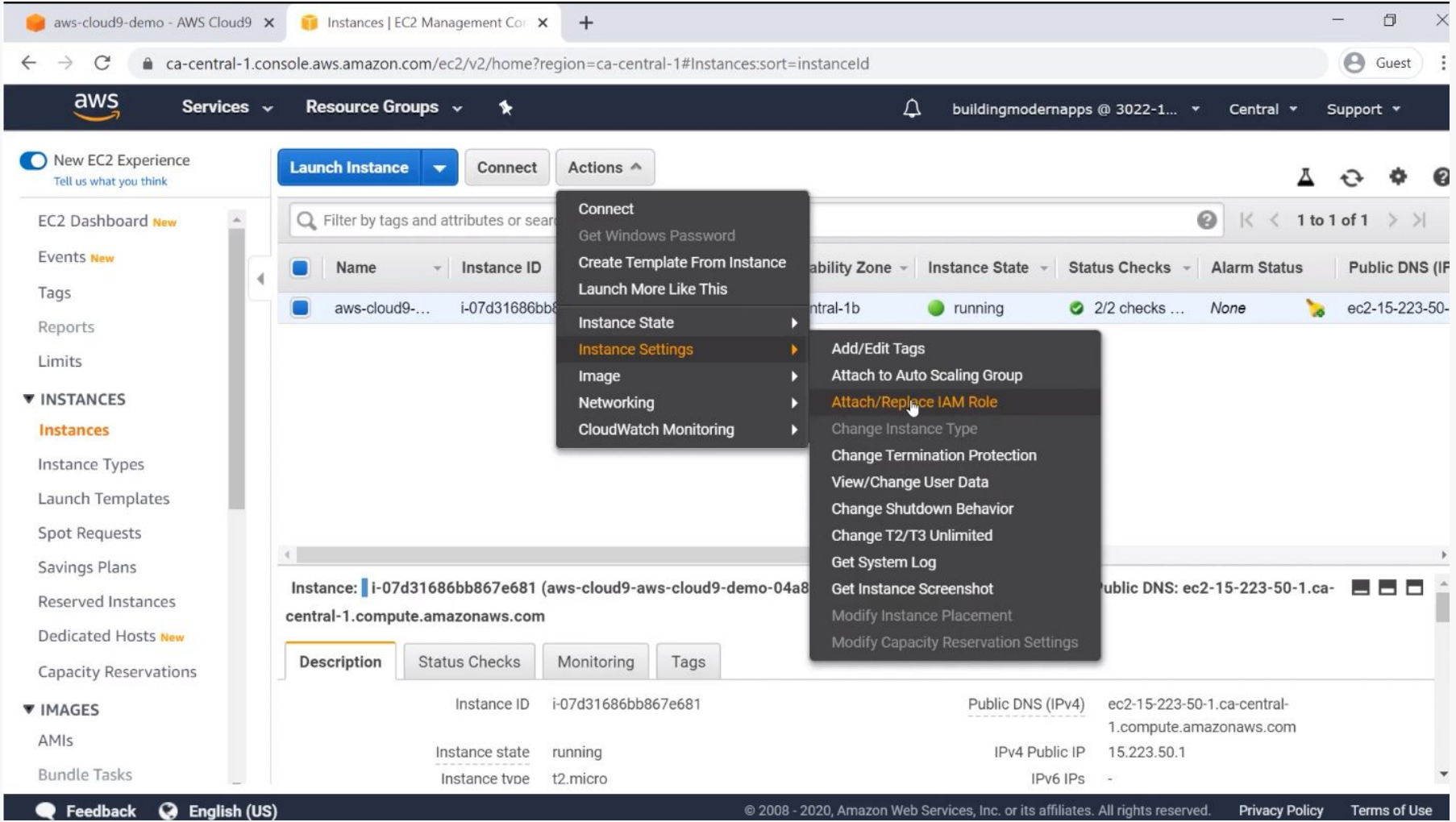

aws-cloud9-demo - AWS Cloud9
Your environments
ca-central-1.console.aws.amazon.com/cloud9/home?region=ca-central-1
Guest
aws
Services
Resource Groups
buildingmodernapps @ 3022-1...
Central
Support
AWS Cloud9
Your environments
Shared with you
Account environments
How-to guide
AWS Cloud9 > Your environments
Your environments (1)
Open IDE
View details
Edit
Delete
Create environment
1
aws-cloud9-demo
Type
EC2
Permissions
Owner
Description
No description available
Owner Arn
arn:aws:iam::302211264422:user/buildingmodernapps
Open IDE
Feedback
English (US)
© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

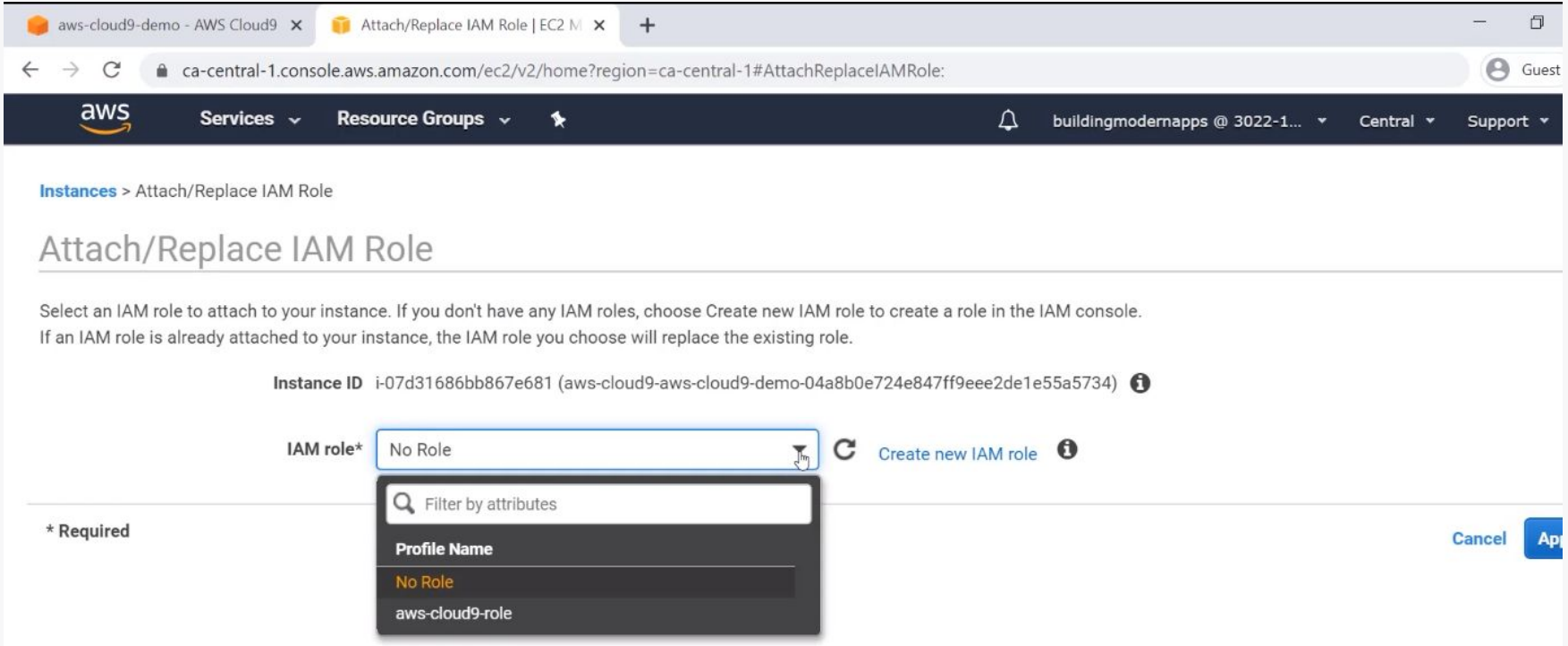
Then under services, click under EC2.
Then navigate to my instances and just click on Instances.

Then find AWS Cloud9 instance which starts with aws-cloud9.

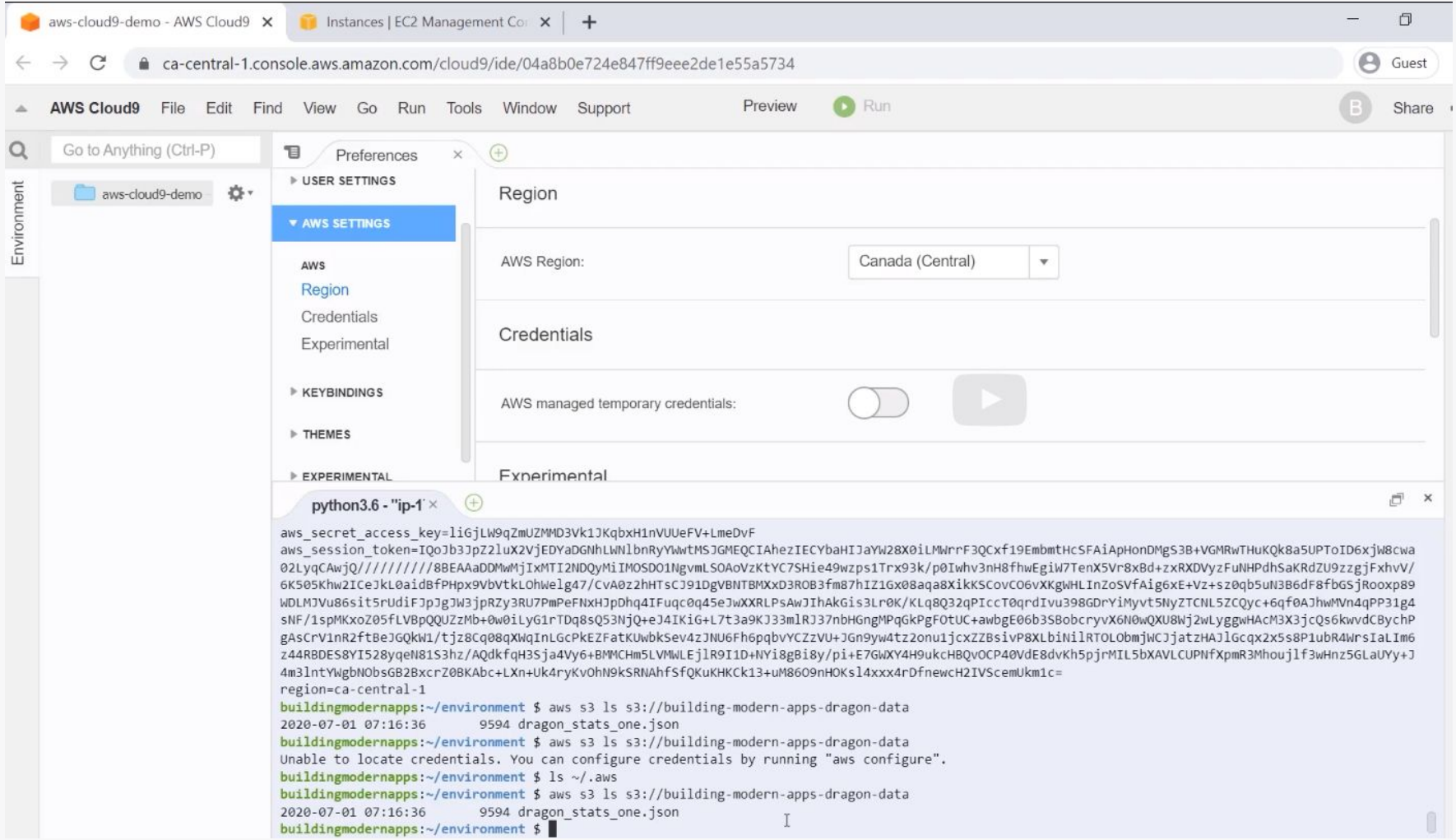
Then click on Actions, Instance Settings, Attach/Replace IAM Role, and then select a role (here I have already created containing the permissions that I'm looking for).


aws-cloud9-demo - AWS Cloud9
Instances | EC2 Management Co
ca-central-1.console.aws.amazon.com/ec2/v2/home?region=ca-central-1#Instances:sort=instanceId
Guest
aws
Services
Resource Groups
buildingmodernapps @ 3022-1...
Central
Support
New EC2 Experience
Tell us what you think
EC2 Dashboard New
Events New
Tags
Reports
Limits
INSTANCES
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts New
Capacity Reservations
IMAGES
AMIs
Bundle Tasks
Launch Instance
Connect
Actions
Filter by tags and attributes or search by keyword
1 to 1 of 1
Name
Instance ID
Instance Type
Availability Zone
Instance State
Status Checks
Alarm Status
Public DNS (IP
aws-cloud9-
i-07d31686bb867e6...
t2.micro
ca-central-1b
running
2/2 checks ...
None
ec2-15-223-50-
Instance: i-07d31686bb867e681 (aws-cloud9-aws-cloud9-demo-04a8b0e724e847ff9eee2de1e55a5734)
Public DNS: ec2-15-223-50-1.ca-central-1.compute.amazonaws.com
Description
Status Checks
Monitoring
Tags
Instance ID
i-07d31686bb867e681
Instance state
running
Instance type
t2.micro
Public DNS (IPv4)
ec2-15-223-50-1.ca-central-1.compute.amazonaws.com
IPv4 Public IP
15.223.50.1
IPv6 IPs
-
Feedback
English (US)
© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use


aws-cloud9-demo - AWS Cloud9
Instances | EC2 Management Co
ca-central-1.console.aws.amazon.com/ec2/v2/home?region=ca-central-1#Instances:sort=instanceId
Guest
aws
Services
Resource Groups
buildingmodernapps @ 3022-1...
Central
Support
New EC2 Experience
Tell us what you think
EC2 Dashboard New
Events New
Tags
Reports
Limits
INSTANCES
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts New
Capacity Reservations
IMAGES
AMIs
Bundle Tasks
Launch Instance
Connect
Actions
Filter by tags and attributes or sear
1 to 1 of 1
Name
Instance ID
ability Zone
Instance State
Status Checks
Alarm Status
Public DNS (IP
aws-cloud9-...
i-07d31686bb8
ntral-1b
running
2/2 checks ...
None
ec2-15-223-50-
Connect
Get Windows Password
Create Template From Instance
Launch More Like This
Instance State
Instance Settings
Image
Networking
CloudWatch Monitoring
Add/Edit Tags
Attach to Auto Scaling Group
Attach/Replace IAM Role
Change Instance Type
Change Termination Protection
View/Change User Data
Change Shutdown Behavior
Change T2/T3 Unlimited
Get System Log
Get Instance Screenshot
Modify Instance Placement
Modify Capacity Reservation Settings
Instance: i-07d31686bb867e681 (aws-cloud9-aws-cloud9-demo-04a8
ublic DNS: ec2-15-223-50-1.ca-central-1.compute.amazonaws.com
Description
Status Checks
Monitoring
Tags
Instance ID
i-07d31686bb867e681
Instance state
running
Instance type
t2.micro
Public DNS (IPv4)
ec2-15-223-50-1.ca-central-1.compute.amazonaws.com
IPv4 Public IP
15.223.50.1
IPv6 IPs
-
Feedback
English (US)
© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

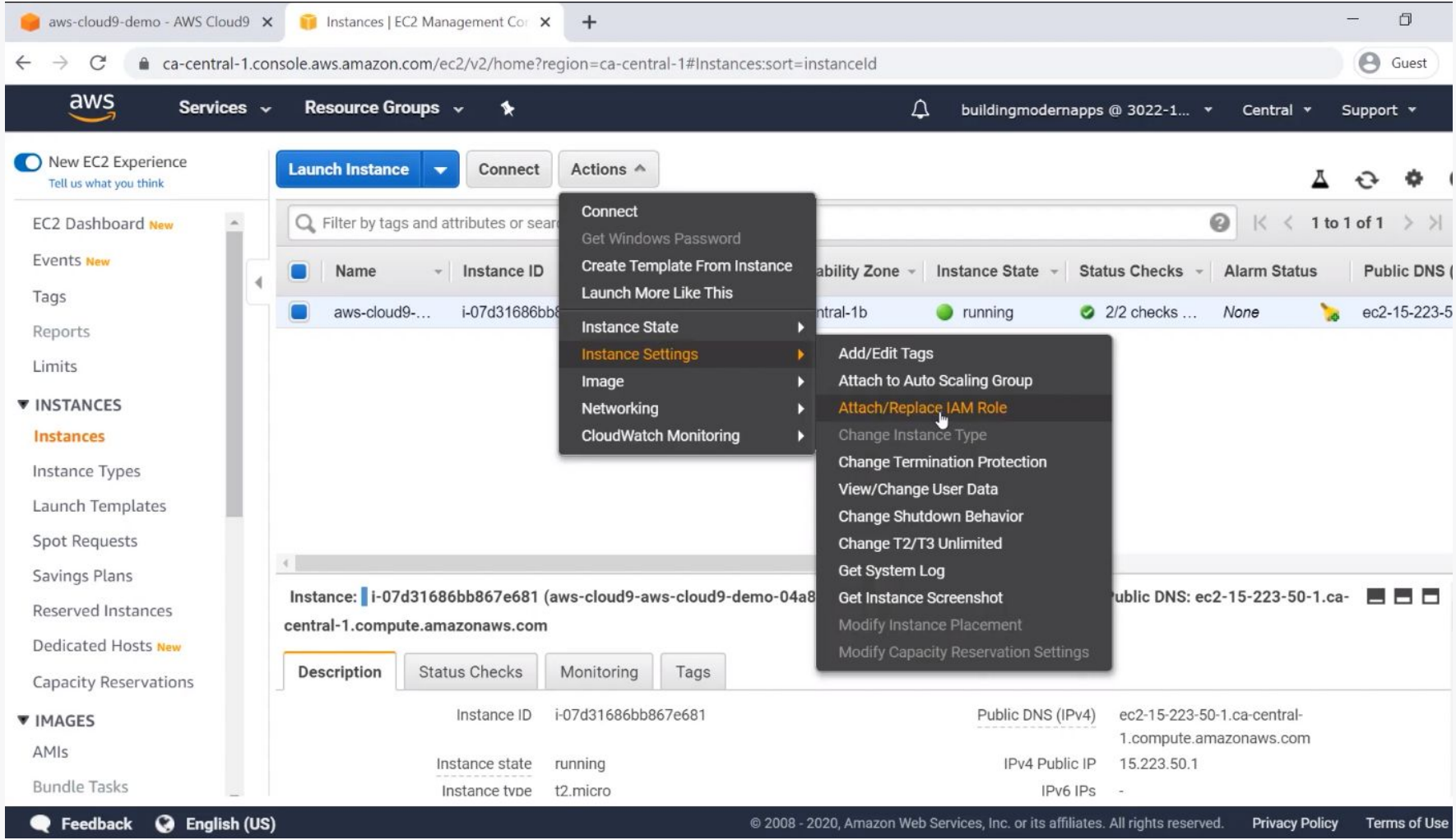

aws-cloud9-demo - AWS Cloud9
Attach/Replace IAM Role | EC2 M
ca-central-1.console.aws.amazon.com/ec2/v2/home?region=ca-central-1#AttachReplaceIAMRole:
Guest
aws
Services
Resource Groups
buildingmodernapps @ 3022-1...
Central
Support
Instances > Attach/Replace IAM Role
Attach/Replace IAM Role
Select an IAM role to attach to your instance. If you don't have any IAM roles, choose Create new IAM role to create a role in the IAM console.
If an IAM role is already attached to your instance, the IAM role you choose will replace the existing role.
Instance ID i-07d31686bb867e681 (aws-cloud9-aws-cloud9-demo-04a8b0e724e847ff9eee2de1e55a5734)
IAM role* No Role
Create new IAM role
Filter by attributes
Profile Name
No Role
aws-cloud9-role
* Required
Cancel

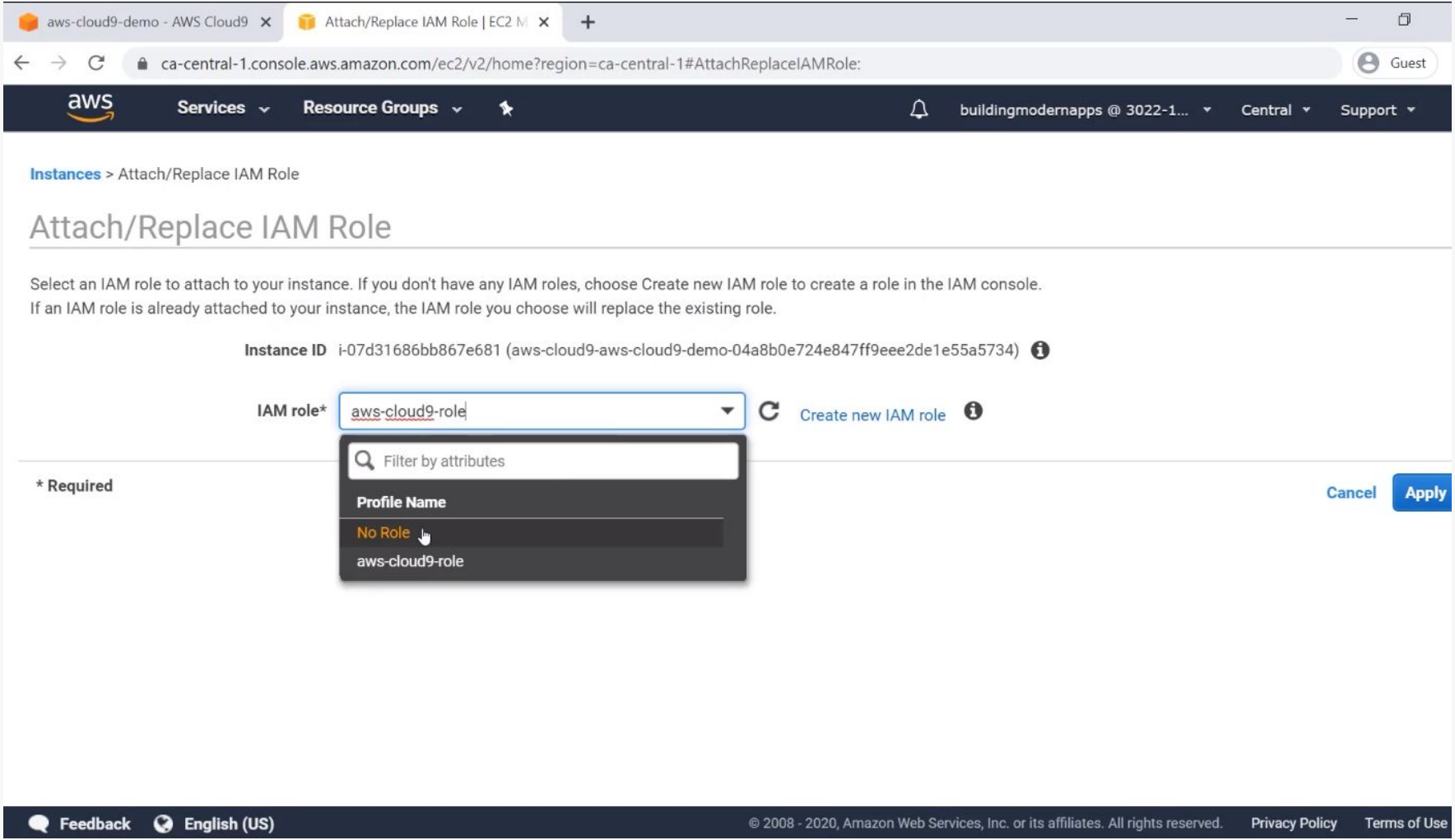
Now, back into the Cloud9 terminal now,
you can execute the exact same command again for listing the bucket.


aws-cloud9-demo - AWS Cloud9
Instances | EC2 Management Co
ca-central-1.console.aws.amazon.com/cloud9/ide/04a8b0e724e847ff9eee2de1e55a5734
Guest
AWS Cloud9 File Edit Find View Go Run Tools Window Support
Preview
Run
Share
Go to Anything (Ctrl-P)
aws-cloud9-demo
Environment
Preferences
USER SETTINGS
AWS SETTINGS
AWS
Region
Credentials
Experimental
KEYBINDINGS
THEMES
EXPERIMENTAL
Region
AWS Region:
Canada (Central)
Credentials
AWS managed temporary credentials:
Experimental
python3.6 - "ip-1
aws_secret_access_key=liGjLW9qZmUZMMD3Vk1JKqbxH1nVUUeFV+LmeDvF
aws_session_token=IQoJb3JpZ2luX2VjEDYaDGNhLWNlbnRyYWwtMSJGMEQCIAhezIECYbaHIJaYW28X0iLMWrrF3QCxf19EmbmtHcSFAiApHonDMgS3B+VGMRwTHuKQk8a5UPToID6xjW8cwa
02LyqCAwjQ/////////8BEAAaDDMwMjIxMTI2NDQyMiIMOSDO1NgvmLSOAoVzKtYC7SHie49wzps1Trx93k/p0Iwhv3nH8fhwEgiW7TenX5Vr8xBd+zxRXDVyzFuNHPdhSaKRdZU9zzgjFxhvV/
6K505Khw2ICeJkL0aidBfPHpx9VbVtkLOhWelg47/CvA0z2hHTsCJ91DgVBNTBMXxD3ROB3fm87hIZ1Gx08aqa8XikKSCovCO6vXKgWHLInZoSVfAig6xE+Vz+sz0qb5uN3B6dF8fbGSjRooxp89
WDLMJVu86sit5rUdiFJpJgJW3jpRZy3RU7PmPeFNxHJpDhq4IFuqc0q45eJwXXRLPsAwJIhAkGis3Lr0K/KLq8Q32qPIccT0qrdIvu398GDrYiMyvt5NyZTCNL5ZCQyc+6qf0AJhwMVn4qPP31g4
sNF/1spMKxoZ05fLVBpQQUZzMb+0w0iLyG1rTDq8sQ53NjQ+eJ4IKiG+L7t3a9KJ33mlRJ37nbHGngMPqGkPgFOtUC+awbgE06b3SBobcryvX6N0wQXU8Wj2wLyggwHAcM3X3jcQs6kwvdCBychP
gAsCrV1nR2ftBeJGQkW1/tjz8Cq08qXWqInLGcPkEZFatKUwbkSev4zJNU6Fh6pqbvYCZzVU+JGn9yw4tz2onu1jcxZZBsivP8XLbiNilRTOLObmjWCJjatzHAJlGcqx2x5s8P1ubR4WrsIaLIm6
z44RBDES8YI528yqeN81S3hz/AQdkfqH3Sja4Vy6+BMMCHm5LVMWLEjlR9I1D+NYi8gBi8y/pi+E7GWXY4H9ukcHBQvOCP40VdE8dvKh5pjrMIL5bXAVLCUPNfXpmR3Mhoujlf3wHnz5GLaUYy+J
4m3lntYWgbNObsGB2BxcrZ0BKAbc+LXn+Uk4ryKvOhN9kSRNAhfSfQKuKHKCk13+uM86O9nHOKsl4xxx4rDfnewcH2IVScemUkm1c=
region=ca-central-1
buildingmodernapps:~/environment $ aws s3 ls s3://building-modern-apps-dragon-data
2020-07-01 07:16:36 9594 dragon_stats_one.json
buildingmodernapps:~/environment $ aws s3 ls s3://building-modern-apps-dragon-data
Unable to locate credentials. You can configure credentials by running "aws configure".
buildingmodernapps:~/environment $ ls ~/.aws
buildingmodernapps:~/environment $ aws s3 ls s3://building-modern-apps-dragon-data
2020-07-01 07:16:36 9594 dragon_stats_one.json
buildingmodernapps:~/environment $

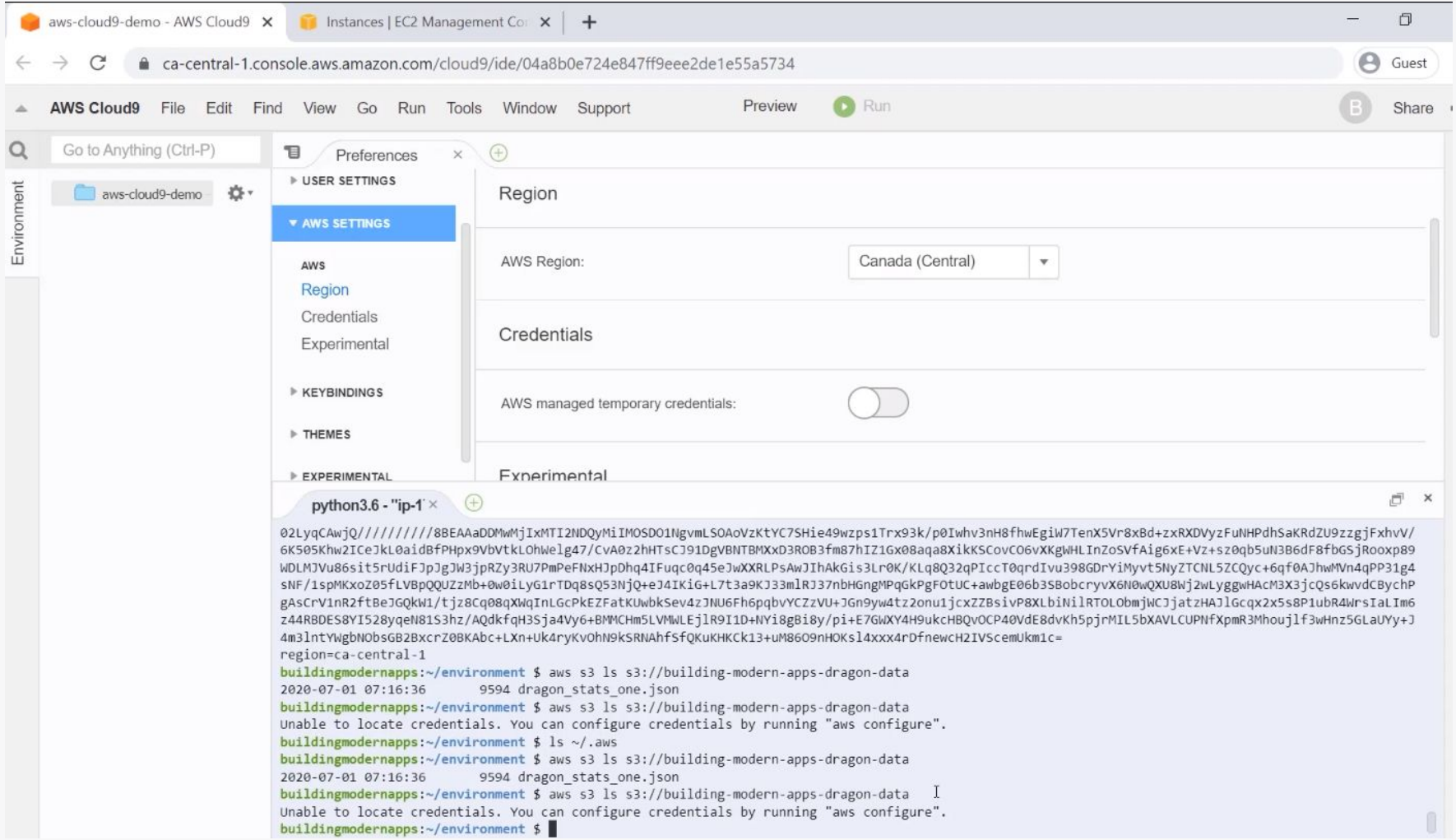
We get the exact same output that I had a little bit earlier, as expected, because we just gave this Cloud9 instance access via an IAM role.

Before discuss the second option,
undo the settings.

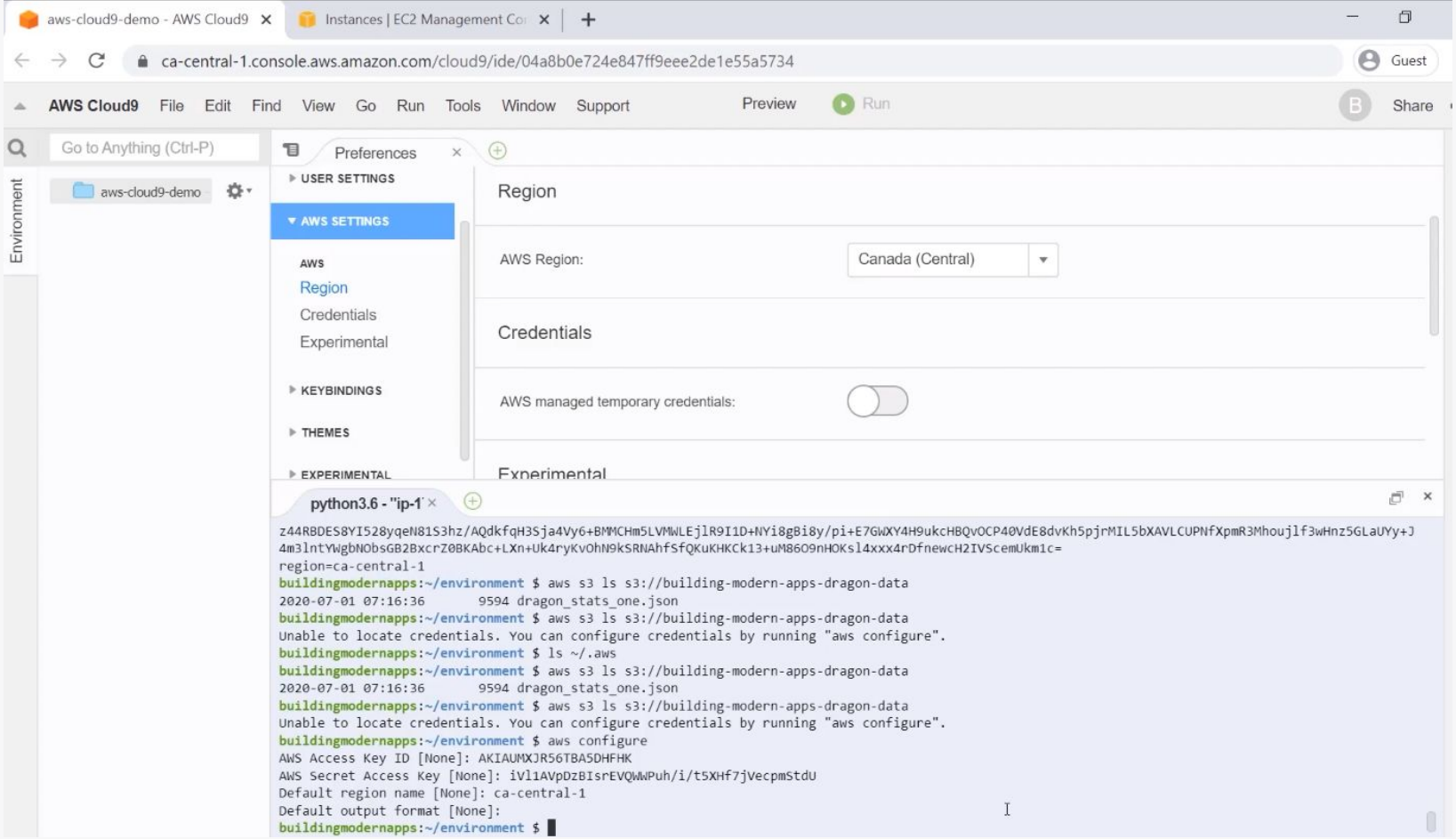

aws-cloud9-demo - AWS Cloud9
Instances | EC2 Management Co
ca-central-1.console.aws.amazon.com/ec2/v2/home?region=ca-central-1#Instances:sort=instanceId
Guest
aws
Services
Resource Groups
buildingmodernapps @ 3022-1...
Central
Support
New EC2 Experience
Tell us what you think
EC2 Dashboard New
Events New
Tags
Reports
Limits
INSTANCES
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts New
Capacity Reservations
IMAGES
AMIs
Bundle Tasks
Launch Instance
Connect
Actions
Connect
Get Windows Password
Create Template From Instance
Launch More Like This
Instance State
Instance Settings
Image
Networking
CloudWatch Monitoring
Add/Edit Tags
Attach to Auto Scaling Group
Attach/Replace IAM Role
Change Instance Type
Change Termination Protection
View/Change User Data
Change Shutdown Behavior
Change T2/T3 Unlimited
Get System Log
Get Instance Screenshot
Modify Instance Placement
Modify Capacity Reservation Settings
Filter by tags and attributes or sear
1 to 1 of 1
Name
Instance ID
ability Zone
Instance State
Status Checks
Alarm Status
Public DNS (
aws-cloud9-...
i-07d31686bb8
ntral-1b
running
2/2 checks ...
None
ec2-15-223-5
Instance: i-07d31686bb867e681 (aws-cloud9-aws-cloud9-demo-04a8
ublic DNS: ec2-15-223-50-1.ca-central-1.compute.amazonaws.com
Description
Status Checks
Monitoring
Tags
Instance ID
i-07d31686bb867e681
Instance state
running
Instance type
t2.micro
Public DNS (IPv4)
ec2-15-223-50-1.ca-central-1.compute.amazonaws.com
IPv4 Public IP
15.223.50.1
IPv6 IPs
-
Feedback
English (US)
© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use


aws-cloud9-demo - AWS Cloud9
Attach/Replace IAM Role | EC2 M
ca-central-1.console.aws.amazon.com/ec2/v2/home?region=ca-central-1#AttachReplaceIAMRole:
Guest
aws
Services
Resource Groups
buildingmodernapps @ 3022-1...
Central
Support
Instances > Attach/Replace IAM Role
Attach/Replace IAM Role
Select an IAM role to attach to your instance. If you don't have any IAM roles, choose Create new IAM role to create a role in the IAM console.
If an IAM role is already attached to your instance, the IAM role you choose will replace the existing role.
Instance ID i-07d31686bb867e681 (aws-cloud9-aws-cloud9-demo-04a8b0e724e847ff9eee2de1e55a5734)
IAM role* aws-cloud9-role
Create new IAM role
Filter by attributes
Profile Name
No Role
aws-cloud9-role
* Required
Cancel
Apply
Feedback
English (US)
© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

If you list the bucket content again,
it will say unable to locate credentials.


aws-cloud9-demo - AWS Cloud9
Instances | EC2 Management Co
ca-central-1.console.aws.amazon.com/cloud9/ide/04a8b0e724e847ff9eee2de1e55a5734
Guest
AWS Cloud9 File Edit Find View Go Run Tools Window Support
Preview
Run
Share
Go to Anything (Ctrl-P)
Environment
aws-cloud9-demo
Preferences
USER SETTINGS
AWS SETTINGS
AWS
Region
Credentials
Experimental
KEYBINDINGS
THEMES
EXPERIMENTAL
Region
AWS Region:
Canada (Central)
Credentials
AWS managed temporary credentials:
Experimental
python3.6 - "ip-1
02LyqCAwjQ/////////8BEAAaDDMwMjIxMTI2NDQyMiIMOSDO1NgvmLSOAoVzKtYC7SHie49wzps1Trx93k/p0Iwhv3nH8fhwEgiW7TenX5Vr8xBd+zxRXDVyzFuNHPdhSaKRdZU9zzgjFxhvV/
6K505Khw2ICeJkL0aidBfPHpx9VbVtkLOhWelg47/CvA0z2hHTsCJ91DgVBNTBMXxD3ROB3fm87hIZ1Gx08aqa8XikKSCovCO6vXKgWHLInZoSVfAig6xE+Vz+sz0qb5uN3B6dF8fbGSjRooxp89
WDLMJVu86sit5rUdiFJpJgJW3jpRZy3RU7PmPeFNxHJpDhq4IFuqc0q45eJwXXRLPsAwJIhAkGis3Lr0K/KLq8Q32qPIccT0qrdIvu398GDrYiMyvt5NyZTCNL5ZCQyc+6qf0AJhwMVn4qPP31g4
sNF/1spMKxoZ05fLVBpQQUZzMb+0w0iLyG1rTDq8sQ53NjQ+eJ4IKiG+L7t3a9KJ33mlRJ37nbHGngMPqGkPgFOtUC+awbgE06b3SBobcryvX6N0wQXU8Wj2wLyggwHAcM3X3jcQs6kwvdCBychP
gAsCrV1nR2ftBeJGQkW1/tjz8Cq08qXWqInLGcPkEZFatKUwbkSev4zJNU6Fh6pqbvYCZzVU+JGn9yw4tz2onu1jcxZZBsivP8XLbiNilRTOLObmjWCJjatzHAJlGcqx2x5s8P1ubR4WrsIaLIm6
z44RBDES8YI528yqeN81S3hz/AQdkfqH3Sja4Vy6+BMMCHm5LVMWLEjlR9I1D+NYi8gBi8y/pi+E7GWXY4H9ukcHBQvOCP40VdE8dvKh5pjrMIL5bXAVLCUPNfXpmR3Mhoujlf3wHnz5GLaUYy+J
4m3lntYWgbNObsGB2BxcrZ0BKAbc+LXn+Uk4ryKvOhN9kSRNAhfSfQKuKHKCk13+uM86O9nHOKsl4xxx4rDfnewcH2IVScemUkm1c=
region=ca-central-1
buildingmodernapps:~/environment $ aws s3 ls s3://building-modern-apps-dragon-data
2020-07-01 07:16:36 9594 dragon_stats_one.json
buildingmodernapps:~/environment $ aws s3 ls s3://building-modern-apps-dragon-data
Unable to locate credentials. You can configure credentials by running "aws configure".
buildingmodernapps:~/environment $ ls ~/.aws
buildingmodernapps:~/environment $ aws s3 ls s3://building-modern-apps-dragon-data
2020-07-01 07:16:36 9594 dragon_stats_one.json
buildingmodernapps:~/environment $ aws s3 ls s3://building-modern-apps-dragon-data
Unable to locate credentials. You can configure credentials by running "aws configure".
buildingmodernapps:~/environment $

The second option to bring credentials within your Cloud9 environment for your code to run is to add an access key and secret key to your credentials file.



I have already generated those credentials for my IAM user.

That's all you need to know about Cloud9 and managed temporary credentials (for now).

If you have issues in the future with certain commands that should be allowed via your IAM user, but aren't allowed in Cloud9, you'll know why.

Serverless Application Model

The AWS Serverless Application Model, or SAM,
is an open source framework for building serverless applications.

It provides shorthand syntax to
express functions, APIs, databases, and more
giving you the ability to define an application you want to model,
using a few short lines per resource.

During deployment, SAM transforms and expands the SAM syntax into AWS CloudFormation syntax, enabling you to build serverless applications faster.

A serverless application in this case is a combination of AWS Lambda functions, event resources, and other resources that work together to perform your distributed tasks.

SAM consists of two main components:

- AWS SAM template specification
- AWS SAM Command Line Interface

The SAM template specification is
used to define the serverless application.

It provides you with the syntax to describe everything
that makes up your serverless application.

The SAM Command Line Interface, or SAM CLI,
is the tool used to build the applications that are defined by the SAM templates.

This provides the commands that enable you to verify that template files are written according to specification, work with resources, package and deploy the application to the AWS cloud, and so on.

Keep in mind that this is a different CLI from the traditionally used AWS CLI, as this is specifically used with SAM.

It needs to be installed and configured separately in order to run.

SAM is an extension of CloudFormation.

Because of that, you get the same deployment capabilities,
you can define resources, use your CloudFormation in your SAM template,
and you can use the full suite of resources, intrinsic functions,
and other template features that are available in CloudFormation.

Because SAM integrates with other AWS services,
there are a lot of benefits that are provided.

Benefits

- Ease of organization: SAM makes it easier to organize related components and resources and operate on a single stack.
- Enforce best practices: SAM enables you to templatize what you're deploying which it makes it possible for you to use and enforce best practices, such as code reviews.
- Ease of deployment: You can use SAM with a suite of AWS tools for building serverless applications.
- Integrated into AWS: You can use the AWS Cloud9 IDE to author, test, and debug your SAM-based serverless applications. You can build a deployment pipeline for your serverless applications using AWS CodeBuild, AWS CodeDeploy, and AWS CodePipeline.

I strongly recommend you continue to dive into the functionality and versatility of the AWS Serverless Application Model.

Additional documentation for you to use as a starting point is uploaded in Moodle.

Lab 1

The labs for this serie of lectures are build out the Dragon APIs.

You will complete the labs in your own AWS account.

In this first lab, you will install and configure the AWS Command Line Interface (AWS CLI) and software development kit (SDK).

After you install the lab requirements, you will create an Amazon Simple Storage Service (Amazon S3) bucket and deploy a web application to the bucket.

Then, you will set up data in the S3 bucket and configure AWS Systems Manager parameters for the Dragons application.

After you create all the resources, you will explore the application.

Lab 1 is a part of a sequence of exercises that guide you through the process of setting up, configuring, securing, deploying, and optimizing a serverless application on AWS.

The final Lab includes instructions to delete all the resources that you created.

Lab 1: Setting Up and Exploring the SDK

https://aws-tc-largeobjects.s3.amazonaws.com/DEV-AWS-MO-BuildingRedux/exercise-1-exploring.html

Here some notes about this lab
that you might find useful.

```
$ echo "Please enter a bucket name: "; read bucket; export MYBUCKET=$bucket
```

- `echo` prints the message "Please enter a bucket name: " to the console.
- `read bucket;` reads input from the user and assigns it to a variable named bucket.
- `export MYBUCKET=$bucket` exports the value stored in the bucket variable to an environment variable named MYBUCKET. This means that MYBUCKET will be available for other commands or scripts to use.

```
$ aws s3 mb s3://$MYBUCKET
```

- `aws` is the AWS Command Line Interface, a tool that allows you to interact with various AWS services from the command line.
- `s3` specifies that you want to interact with the Simple Storage Service (S3).
- `mb` stands for "make bucket". It's a command used to create a new S3 bucket.
- `s3://$MYBUCKET` is a URL-like notation used by AWS to refer to S3 buckets. $MYBUCKET is a variable that holds the name of the bucket. This variable is assumed to be previously defined or set in your environment.

```
echo "export MYBUCKET=$MYBUCKET" >> ~/.bashrc
```

- The command is adding a line to the .bashrc file that sets the environment variable MYBUCKET to its current value. This can be useful if you want to persistently keep the MYBUCKET environment variable across shell sessions.

```
echo "export MYBUCKET=$MYBUCKET" >> ~/.bashrc
```

- `echo "export MYBUCKET=$MYBUCKET` prints the string export MYBUCKET=$MYBUCKET
- `>> ~/.bashrc` appends the output of the previous echo command to the file ~/.bashrc. The >> operator is used for appending text to a file.
- `~` is a shorthand notation for the user's home directory
- `~/.bashrc` is the file that is being modified. It's a hidden file in the user's home directory that is read by the Bash shell when it starts. It's typically used for setting environment variables and customizing the behavior of the shell.

```
aws s3 ls
```

- This command list the contents of your S3 buckets.

```
aws s3api put-bucket-ownership-controls --bucket $MYBUCKET --ownership-controls
                'Rules=[{ObjectOwnership=BucketOwnerPreferred}]'
```

- `s3api`
  - a sub-command for interacting with S3 using a more direct API approach
- `put-bucket-ownership-controls`
  - This is the specific command to put (i.e., configure) the bucket ownership controls for a given S3 bucket.
- `--ownership-controls 'Rules=[{ObjectOwnership=BucketOwnerPreferred}]'`
  - This is the ownership control configuration being applied to the bucket. It's using a JSON-like syntax to specify the rules. In this case, it's setting the rule that object ownership should be "BucketOwnerPreferred". This means that if an object is uploaded without a specific owner, the bucket owner will be considered the owner by default.

The Public Access Block is a feature provided by Amazon S3 (Simple Storage Service) that allows you to manage public access to your S3 buckets and their objects.

It helps prevent accidental exposure of your data to the public.

```
aws s3api put-public-access-block --bucket $MYBUCKET --public-access-block-configuration
"BlockPublicAcls=false,IgnorePublicAcls=false,BlockPublicPolicy=false,RestrictPublicBuckets=false"
```

- `put-public-access-block`
  - a specific command to put (i.e., configure) the public access block settings for a given S3 bucket.
- `--public-access-block-configuration`
  - is the configuration being applied. It's using a string with comma-separated key-value pairs to specify the settings. In this case, it's setting the following parameters:
    - BlockPublicAcls=false: This means that public ACLs (Access Control Lists) are not blocked.
    - IgnorePublicAcls=false: This means that public ACLs are not ignored.
    - BlockPublicPolicy=false: This means that public policies are not blocked.
    - RestrictPublicBuckets=false: This means that public access to buckets is not restricted.

When using the AWS CLI, you can specify configurations for S3 bucket settings in two formats: JSON Format and Key-Value String Pairs.

```
aws s3api put-public-access-block --bucket $MYBUCKET --public-access-block-configuration '{
  "BlockPublicAcls": false,
  "IgnorePublicAcls": false,
  "BlockPublicPolicy": false,
  "RestrictPublicBuckets": false
}'
```

is the same as:

```
aws s3api put-public-access-block --bucket $MYBUCKET --public-access-block-configuration
"BlockPublicAcls=false,IgnorePublicAcls=false,BlockPublicPolicy=false,RestrictPublicBuckets=false"
```

- `--recursive`
  - This flag specifies that the command should operate recursively, meaning it will copy all files and directories within the source directory.
- `--acl public-read`
  - This sets the access control list (ACL) of the copied objects to be publicly readable. This means that anyone with the appropriate URL can access the objects in the S3 bucket.

```
aws ssm put-parameter \
--name "dragon_data_bucket_name" \
--type "String" \
--overwrite \
--value $MYBUCKET
```

- in summary, this command is creating a parameter in the SSM Parameter Store named "dragon_data_bucket_name" with the value stored in the MYBUCKET environment variable. This parameter could be used by other AWS services or applications to access or reference the S3 bucket.

By using AWS SSM Parameter Store to store configuration values like bucket names or file names, you can decouple your application's configuration from its source code.

So, in your app.py, you would retrieve the values from Parameter Store using the appropriate API or SDK provided by AWS. This allows your app to adapt to changes in configuration without requiring code modifications or redeployment.