

boto3 aws sdk Cheatsheet

Connection

```
import boto3
    Import the boto3 library

boto3.setup_default_session(region_name='us-west-2')
    Set default AWS region

boto3.setup_default_session(profile_name='prod')
    Use a specific AWS profile

boto3.client('s3', aws_access_key_id='KEY',
aws_secret_access_key='SECRET')
    Create client with explicit credentials
```

S3 Operations

```
s3 = boto3.resource('s3')
    Create S3 resource

s3_client = boto3.client('s3')
    Create S3 client

s3_client.list_buckets()
    List all S3 buckets

s3_client.list_objects_v2(Bucket='bucket-name')
    List objects in bucket

s3_client.upload_file('file.txt', 'bucket-name', 'key')
    Upload file to S3

s3_client.download_file('bucket-name', 'key', 'file.txt')
    Download file from S3

s3_client.delete_object(Bucket='bucket-name', Key='key')
    Delete object from S3
```

EC2 Operations

```
ec2 = boto3.resource('ec2')
    Create EC2 resource

ec2_client = boto3.client('ec2')
    Create EC2 client

ec2_client.describe_instances()
    List EC2 instances

ec2_client.start_instances(InstanceIds=['i-123456'])
    Start EC2 instance

ec2_client.stop_instances(InstanceIds=['i-123456'])
    Stop EC2 instance

ec2_client.create_tags(Resources=['i-123456'],
Tags=[{'Key': 'Name', 'Value': 'server'}])
    Add tags to EC2 instance
```

CloudWatch Operations

```
cloudwatch = boto3.client('cloudwatch')
    Create CloudWatch client

cloudwatch.get_metric_data(MetricDataQueries=[...],
StartTime=..., EndTime=...)
    Get metric data

cloudwatch.put_metric_alarm(AlarmName='...', ...)
    Create CloudWatch alarm

logs = boto3.client('logs')
    Create CloudWatch Logs client

logs.describe_log_groups()
    List log groups

logs.get_log_events(logGroupName='...', logStreamName='...')
    Get log events
```

Lambda Operations

```
lambda_client = boto3.client('lambda')
    Create Lambda client

lambda_client.list_functions()
    List Lambda functions

lambda_client.invoke(FunctionName='name', Payload=b'{}')
    Invoke Lambda function

lambda_client.get_function(FunctionName='name')
    Get Lambda function details

lambda_client.update_function_code(FunctionName='name', ZipFile=bytes)
    Update Lambda function code
```

Error Handling

```
try: ... except botocore.exceptions.ClientError as e:
    print(e)
    Basic error handling

from botocore.exceptions import ClientError
    Import specific exception

response = ... if response['ResponseMetadata']['HTTPStatusCode'] == 200: ...
    Check response status
```

Paginate

```
paginator = s3_client.get_paginator('list_objects_v2')
    Create paginator

for page in paginator.paginate(Bucket='bucket-name'): ...
    Use paginator to iterate through pages
```

SNS/SQS Operations

```
sns = boto3.client('sns')
    Create SNS client

sns.list_topics()
    List SNS topics

sns.publish(TopicArn='arn', Message='message')
    Publish message to SNS topic

sqs = boto3.client('sqs')
    Create SQS client

sqs.list_queues()
    List SQS queues

sqs.receive_message(QueueUrl='url')
    Receive messages from SQS queue

sqs.send_message(QueueUrl='url', MessageBody='message')
    Send message to SQS queue
```

DynamoDB Operations

```
dynamodb = boto3.resource('dynamodb')
    Create DynamoDB resource

dynamodb_client = boto3.client('dynamodb')
    Create DynamoDB client

dynamodb_client.list_tables()
    List DynamoDB tables

table = dynamodb.Table('table-name')
    Get DynamoDB table

table.put_item(Item={'id': '1', 'name': 'test'})
    Insert item into table

table.get_item(Key={'id': '1'})
    Get item from table

table.query(KeyConditionExpression=Key('id').eq('1'))
    Query items from table
```

Waiters

```
waiter = ec2_client.get_waiter('instance_running')
    Create waiter

waiter.wait(InstanceIds=['i-123456'])
    Wait for instance to be running
```