

1:- Create IAM Role

The screenshot shows the AWS IAM Roles page. The left sidebar has sections for Dashboard, Access management (User groups, Users, Roles, Policies, Identity providers, Account settings, Root access management), and Access reports. The main area is titled 'Permissions' and shows 'Permissions policies (5)'. It lists five AWS managed policies: AmazonEC2FullAccess (121 attached entities), AmazonSNSFullAccess (37), AWSLambdaBasicExecutionRole (73), CloudWatchReadOnlyAccess (7), and ElasticLoadBalancingReadOnly (1). There are tabs for Trust relationships, Tags, Last Accessed, and Revoke sessions. A search bar and filter by type are at the top. The bottom right corner includes copyright information and links for Privacy, Terms, and Cookie preferences.

2:- Create EC2 Instance

The screenshot shows the AWS EC2 Instances Launch an instance page. The left sidebar has sections for Recents, My AMIs, and Quick Start. The main area is titled 'Launch an instance' and shows fields for Name and tags (Name: 'adish-web'), Application and OS Images (Amazon Machine Image) search, and a summary section. The summary includes Number of instances (1), Software Image (AMI) (Amazon Linux 2023 AMI 2023.9.2...), Virtual server type (instance type) (t2.micro), Firewall (security group) (New security group), and a 'Launch instance' button. The bottom right corner includes copyright information and links for Privacy, Terms, and Cookie preferences.

The screenshot shows the AWS EC2 Instances Launch an instance page with detailed configuration options. The left sidebar has sections for Recents, My AMIs, and Quick Start. The main area includes sections for Instance type (t2.micro, Family: t2, 1 vCPU, 1 GiB Memory, Current generation: true, Free tier eligible, On-Demand Ubuntu Pro base pricing: 0.015 USD per Hour, On-Demand Windows base pricing: 0.0178 USD per Hour, On-Demand RHEL base pricing: 0.0276 USD per Hour, On-Demand SUSE base pricing: 0.0132 USD per Hour, On-Demand Linux base pricing: 0.0132 USD per Hour), Additional costs apply for AMIs with pre-installed software, Key pair (login) (Key pair name: 'adish-MERN-backend', Create new key pair), and a Summary section. The Summary includes Number of instances (1), Software Image (AMI) (Amazon Linux 2023 AMI 2023.9.2...), Virtual server type (instance type) (t2.micro), Firewall (security group) (New security group), and a 'Launch instance' button. The bottom right corner includes copyright information and links for Privacy, Terms, and Cookie preferences.

EC2 > Instances > Launch an instance

Firewall (security groups) | Info
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

Create security group Select existing security group

Common security groups | Info
Select security groups

rajan-sg sg-0bd26d2dc9b46983b X
VPC: vpc-0376eb6043cd8004

Compare security group rules

Security groups that you add or remove here will be added to or removed from all your network interfaces.

Configure storage | Info
Advanced

1x 8 GiB gp3 Root volume, 3000 IOPS, Not encrypted

Add new volume

Summary
Number of instances | Info
1

Software Image (AMI)
Amazon Linux 2023 AMI 2023.9.2...read more
ami-06a8ca19af7f6f3f4

Virtual server type (instance type)
t2.micro

Firewall (security group)
rajan-sg

Cancel Launch instance Preview code

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

EC2 > Instances > i-02887b366874a50f6

Instance summary for i-02887b366874a50f6 (adish_web) | Info

Updated less than a minute ago

| | | |
|---|--|--|
| Instance ID i-02887b366874a50f6 | Public IPv4 address 18.134.3.169 open address | Private IPv4 addresses 172.31.12.14 |
| IPv6 address - | Instance state Running | Public DNS ec2-18-134-3-169.eu-west-2.compute.amazonaws.com open address |
| Hostname type IP name: ip-172-31-12-14.eu-west-2.compute.internal | Private IP DNS name (IPv4 only) ip-172-31-12-14.eu-west-2.compute.internal | Elastic IP addresses - |
| Answer private resource DNS name IPv4 (A) | Instance type t2.micro | AWS Compute Optimizer finding |
| Auto-assigned IP address 18.134.3.169 [Public IP] | VPC ID vpc-0376eb6043cd8004 | User: arn:aws:iam::975050024946:user/adishansari |

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

3:- Create SNS

Amazon SNS > Topics > adish-invoice-topic

adish-invoice-topic

Details

| | |
|--|------------------------------------|
| Name adish-invoice-topic | Display name - |
| ARN arn:aws:sns:eu-west-2:975050024946:adish-invoice-topic | Topic owner 975050024946 |
| Type Standard | |

Subscriptions | Access policy | Data protection policy | Delivery policy (HTTP/S) | Delivery status logging

Subscriptions (2)

Edit Delete Request confirmation Confirm subscription Create subscription

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

4 :- Create Lambda Function

Lambda > Functions > Create function

Create function Info

Choose one of the following options to create your function.

- Author from scratch
Start with a simple Hello World example.
- Use a blueprint
Build a Lambda application from sample code and configuration presets for common use cases.
- Container image
Select a container image to deploy for your function.

Basic information

Function name
Enter a name that describes the purpose of your function.

Function name must be 1 to 64 characters, must be unique to the Region, and can't include spaces. Valid characters are a-z, A-Z, 0-9, hyphens (-), and underscores (_).

Runtime Info
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.
 ▼ (C)

Architecture Info
Choose the instruction set architecture you want for your function code.

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Lambda > Functions > Create function

▼ Change default execution role

Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

- Create a new role with basic Lambda permissions
- Use an existing role
- Create a new role from AWS policy templates

Existing role
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.
 ▼ (C)

View the [Swetabh-Lambda_EC2_AutoScaler_Role](#) role on the IAM console.

► Additional configurations

Use additional configurations to set up networking, security, and governance for your function. These settings help secure and customize your Lambda function deployment.

Cancel Create function

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Lambda > Functions > Auto-ScaleEC2InstancesAdish

Code Test Monitor Configuration Aliases Versions

General configuration Info

| | | | |
|--------------------|-------------------------------|--|------------------------------------|
| Triggers | Description - | Memory 128 MB | Ephemeral storage 512 MB |
| Permissions | Timeout 1 min 3 sec | SnapStart <small>Info</small> None | Edit |

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Environment variables

You can define environment variables as key-value pairs that are accessible from your function code. These are useful to store configuration settings without the need to change function code.

[Learn more](#)

| Key | Value | Remove |
|-----------------------|--|-------------------------|
| ELB_NAME | arn:aws:elasticloadbalancing:eu-west-2:975050024946:loadbalancer | <button>Remove</button> |
| METRIC_NAMESPACE | auto-scale-alb | <button>Remove</button> |
| METRIC_NAME | ActiveConnectionCount | <button>Remove</button> |
| METRIC_DIMENSION_NAME | LoadBalancerName | <button>Remove</button> |
| AGGREGATION | Average | <button>Remove</button> |
| PERIOD_SECONDS | 300 | <button>Remove</button> |
| HIGH_THRESHOLD | 80 | <button>Remove</button> |

[CloudShell](#) [Feedback](#)

© 2025, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

Environment variables

| | | |
|--------------------|--|-------------------------|
| LOW_THRESHOLD | 20 | <button>Remove</button> |
| SNS_TOPIC_ARN | arn:aws:sns:eu-west-2:975050024946:adish-invoice-topic | <button>Remove</button> |
| AMI_ID | ami-0f7b02bb6a0e14062 | <button>Remove</button> |
| INSTANCE_TYPE | t2.micro | <button>Remove</button> |
| KEY_NAME | Adish-MERN-Frontend | <button>Remove</button> |
| SECURITY_GROUP_IDS | sg-039ebcd93a99f5586 | <button>Remove</button> |
| SUBNET_ID | subnet-01d94082ca6384cae | <button>Remove</button> |
| MIN_INSTANCES | 3 | <button>Remove</button> |
| MAX_INSTANCES | 7 | <button>Remove</button> |

[CloudShell](#) [Feedback](#)

© 2025, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

Environment variables

| | | |
|--------------------|--------------------------|-------------------------|
| SUBNET_ID | subnet-01d94082ca6384cae | <button>Remove</button> |
| MIN_INSTANCES | 3 | <button>Remove</button> |
| MAX_INSTANCES | 7 | <button>Remove</button> |
| INSTANCE_TAG_KEY | Name | <button>Remove</button> |
| INSTANCE_TAG_VALUE | adish_web | <button>Remove</button> |

[Add environment variable](#)

▶ Encryption configuration

[Cancel](#) [Save](#)

[CloudShell](#) [Feedback](#)

© 2025, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

5:- Create ALB

EC2 > Load balancers > Create Application Load Balancer

Create Application Load Balancer Info

The Application Load Balancer distributes incoming HTTP and HTTPS traffic across multiple targets such as Amazon EC2 instances, microservices, and containers, based on request attributes. When the load balancer receives a connection request, it evaluates the listener rules in priority order to determine which rule to apply, and if applicable, it selects a target from the target group for the rule action.

How Application Load Balancers work

Basic configuration

Load balancer name
Name must be unique within your AWS account and can't be changed after the load balancer is created.

A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen.

Scheme Info
Scheme can't be changed after the load balancer is created.

Internet-facing

- Serves internet-facing traffic.
- Has public IP addresses.
- DNS name resolves to public IPs.
- Compatible with the IPv4 and Dualstack IP address types.

 Internal

- Serves internal traffic.
- Has private IP addresses.
- DNS name resolves to private IPs.

CloudShell Feedback

© 2025, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

EC2 > Load balancers > Create Application Load Balancer

Network mapping Info

The load balancer routes traffic to targets in the selected subnets, and in accordance with your IP address settings.

VPC Info
The load balancer will exist and scale within the selected VPC. The selected VPC is also where the load balancer targets must be hosted unless routing to Lambda or on-premises targets, or if using VPC peering. To confirm the VPC for your targets, view [target groups](#).

vpc-0376ebe6043cd8004 (default)

Use IPAM pool for public IPv4 addresses
The IPAM pool you choose will be the preferred source of public IPv4 addresses. If the pool is depleted IPv4 addresses will be assigned by AWS.

Public IPv4 IPAM pool

The locale of the IPAM pool must be equal to the current Region and the awsService attribute must be EC2.

Availability Zones and subnets Info
Select at least two Availability Zones and a subnet for each zone. A load balancer node will be placed in each selected zone and will automatically scale in response to traffic. The load balancer routes traffic to targets in the selected Availability Zones only.

eu-west-2a (euw2-az2)

CloudShell Feedback

© 2025, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

EC2 > Load balancers > Create Application Load Balancer

Default action Info

The default action is used if no other rules apply. Choose the default action for traffic on this listener.

Routing action
 Forward to target groups Redirect to URL Return fixed response

Forward to target group Info
Choose a target group and specify routing weight or [create target group](#).

| Target group | Weight | Percent |
|------------------------------------|--------|---------|
| adishwebserver <small>HTTP</small> | 1 | 100% |

You can add up to 4 more target groups.

Target group stickiness Info
Enables the load balancer to bind a user's session to a specific target group. To use stickiness the client must support cookies. If you want to bind a user's session to a specific target, turn on the Target Group attribute Stickiness.

Turn on target group stickiness

CloudShell Feedback

© 2025, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

```
import os
import boto3
import logging
from datetime import datetime, timedelta
from botocore.exceptions import ClientError

logger = logging.getLogger()
logger.setLevel(logging.INFO)

# clients
cw = boto3.client('cloudwatch')
ec2 = boto3.client('ec2')
sns = boto3.client('sns')

# Environment variables (set these in Lambda configuration)
ELB_NAME = os.environ.get('ELB_NAME')    # e.g.
"app/my-alb/1234567890abcdef"
METRIC_NAMESPACE = os.environ.get('METRIC_NAMESPACE',
'AWS/ApplicationELB')    # default for ALB
METRIC_NAME = os.environ.get('METRIC_NAME', 'RequestCount')    # or
'ActiveConnectionCount' for NLB, etc.
METRIC_DIMENSION_NAME = os.environ.get('METRIC_DIMENSION_NAME',
'LoadBalancer')    # for AWS/ApplicationELB
HIGH_THRESHOLD = float(os.environ.get('HIGH_THRESHOLD', '80'))    # percent
or absolute depending on metric
LOW_THRESHOLD = float(os.environ.get('LOW_THRESHOLD', '20'))
# Interpret thresholds relative to "per-instance" request capacity when
using RequestCount.
AGGREGATION = os.environ.get('AGGREGATION', 'Average')    # Average, Sum,
Maximum
PERIOD_SECONDS = int(os.environ.get('PERIOD_SECONDS', '300'))    # 5 minutes
SNS_TOPIC_ARN = os.environ.get('SNS_TOPIC_ARN')

# If you want Lambda to create instances directly:
AMI_ID = os.environ.get('AMI_ID')    # e.g. "ami-0123456789abcdef0"
INSTANCE_TYPE = os.environ.get('INSTANCE_TYPE', 't3.micro')
KEY_NAME = os.environ.get('KEY_NAME')    # optional
SECURITY_GROUP_IDS = os.environ.get('SECURITY_GROUP_IDS')    #
comma-separated ids
```

```

SUBNET_ID = os.environ.get('SUBNET_ID')    # subnet to launch into
(optional)
MIN_INSTANCES = int(os.environ.get('MIN_INSTANCES', '1'))    # do not go
below
MAX_INSTANCES = int(os.environ.get('MAX_INSTANCES', '5'))    # cap max
INSTANCE_TAG_KEY = os.environ.get('INSTANCE_TAG_KEY', 'AutoScaleManaged')
INSTANCE_TAG_VALUE = os.environ.get('INSTANCE_TAG_VALUE', 'true')

# Which instances to count/manage: filter by tag.
MANAGED_FILTERS = [{ 'Name': f'tag:{INSTANCE_TAG_KEY}', 'Values':
[INSTANCE_TAG_VALUE] }]

def publish_sns(subject: str, message: str):
    if not SNS_TOPIC_ARN:
        logger.warning("SNS_TOPIC_ARN not configured; skipping
notification.")
        return
    try:
        sns.publish(TopicArn=SNS_TOPIC_ARN, Subject=subject,
Message=message)
        logger.info("SNS published: %s", subject)
    except ClientError as e:
        logger.exception("Failed to publish SNS: %s", e)

def get_metric_value():
    """
    Fetch metric from CloudWatch for the last PERIOD_SECONDS window.
    Returns the requested statistic value (float) or None.
    """
    end = datetime.utcnow()
    start = end - timedelta(seconds=PERIOD_SECONDS)
    dims = [ { 'Name': METRIC_DIMENSION_NAME, 'Value': ELB_NAME } ]
    try:
        resp = cw.get_metric_statistics(
            Namespace=METRIC_NAMESPACE,
            MetricName=METRIC_NAME,
            Dimensions=dims,
            StartTime=start,
            EndTime=end,
            Period=PERIOD_SECONDS,

```

```

        Statistics=[AGGREGATION]
    )
    datapoints = resp.get('Datapoints', [])
    if not datapoints:
        logger.info("No datapoints returned for metric query.")
        return None
    # pick the latest datapoint by timestamp
    latest = max(datapoints, key=lambda d: d['Timestamp'])
    value = latest.get(AGGREGATION)
    logger.info("Metric fetched: %s=%s (timestamp=%s)", METRIC_NAME,
value, latest.get('Timestamp'))
    return float(value)
except ClientError:
    logger.exception("Error fetching metric")
    return None

def count_managed_instances():
    """Count running/stopped instances that match our tag filters (and are
EC2-managed by this function)."""
    try:
        resp = ec2.describe_instances(Filters=MANAGED_FILTERS + [{ 'Name' :
'instance-state-name', 'Values':
['pending','running','stopping','stopped']}])
        instances = []
        for r in resp.get('Reservations', []):
            for i in r.get('Instances', []):
                instances.append(i)
        logger.info("Found %d managed instances", len(instances))
        return instances
    except ClientError:
        logger.exception("Failed to describe instances")
        return []

def start_new_instance():
    """Start a single new EC2 instance, tagged so it is managed."""
    if not AMI_ID:
        raise RuntimeError("AMI_ID not configured; cannot launch
instance.")
    sg_ids = [x.strip() for x in SECURITY_GROUP_IDS.split(',') if
SECURITY_GROUP_IDS else None

```

```

launch_args = {
    'ImageId': AMI_ID,
    'InstanceType': INSTANCE_TYPE,
    'MinCount': 1,
    'MaxCount': 1,
    'TagSpecifications': [
        {
            'ResourceType': 'instance',
            'Tags': [ {'Key': INSTANCE_TAG_KEY, 'Value':
INSTANCE_TAG_VALUE} ]
        }
    ]
}

if sg_ids:
    launch_args['SecurityGroupIds'] = sg_ids
if KEY_NAME:
    launch_args['KeyName'] = KEY_NAME
if SUBNET_ID:
    launch_args['SubnetId'] = SUBNET_ID

try:
    resp = ec2.run_instances(**launch_args)
    instance_id = resp['Instances'][0]['InstanceId']
    logger.info("Launched instance %s", instance_id)
    publish_sns("Scale Up: Launched EC2", f"Launched instance
{instance_id} due to high load.")
    return instance_id
except ClientError:
    logger.exception("Failed to launch instance")
    return None

def terminate_one_instance(instances):
    """
    Terminate one instance from the provided list (prefer stopped or
    oldest running).

    Instances is list of instance dicts from describe_instances.
    """
    # select candidate: prefer instances in 'stopped', then oldest
    'running'
    stopped = [i for i in instances if i['State']['Name'] == 'stopped']
    if stopped:
        target = stopped[0]

```

```

else:
    # pick oldest running (by LaunchTime)
    running = [i for i in instances if i['State']['Name'] ==
'running']
    if not running:
        logger.info("No running instances to terminate.")
        return None
    target = min(running, key=lambda i: i['LaunchTime'])

inst_id = target['InstanceId']
try:
    ec2.terminate_instances(InstanceIds=[inst_id])
    logger.info("Terminated instance %s", inst_id)
    publish_sns("Scale Down: Terminated EC2", f"Terminated instance {inst_id} due to low load.")
    return inst_id
except ClientError:
    logger.exception("Failed to terminate instance")
    return None

def lambda_handler(event, context):
    logger.info("Auto-scale Lambda started")
    metric_value = get_metric_value()
    if metric_value is None:
        logger.warning("Metric not available; exiting without action.")
        return {"status": "no_metric"}

    # You must decide what metric_value represents: percent or absolute.
    # This sample assumes metric is percent-like (0-100). If metric is
    RequestCount, you may need to normalize to per-instance rate.

    try:
        managed = count_managed_instances()
        current_count = len([i for i in managed if i['State']['Name'] in
('pending', 'running', 'stopping', 'stopped')])
        logger.info("Current managed instance count: %d", current_count)
    except Exception as e:
        logger.exception("Failed to count managed instances: %s", e)
        current_count = 0

    # Scaling decisions

```

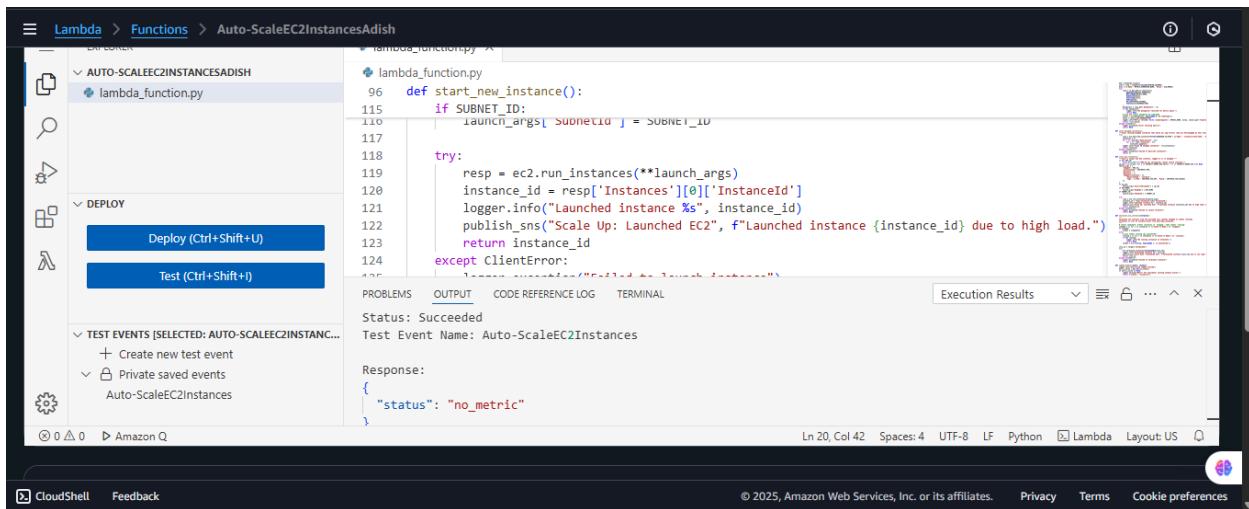
```

# Scale up
if metric_value >= HIGH_THRESHOLD and current_count < MAX_INSTANCES:
    logger.info("High threshold exceeded: %s >= %s", metric_value,
HIGH_THRESHOLD)
    launched = start_new_instance()
    return {"action": "scale_up", "launched": launched}

# Scale down
if metric_value <= LOW_THRESHOLD and current_count > MIN_INSTANCES:
    logger.info("Low threshold met: %s <= %s", metric_value,
LOW_THRESHOLD)
    terminated = terminate_one_instance(managed)
    return {"action": "scale_down", "terminated": terminated}

logger.info("No scaling action required. metric=%s thresholds=(%s,%s)
count=%d", metric_value, LOW_THRESHOLD, HIGH_THRESHOLD, current_count)
return {"action": "no_action", "metric": metric_value}

```



The screenshot shows the AWS Lambda Function Editor interface. On the left, the function structure is displayed with a file tree containing 'lambda_function.py' under 'AUTO-SCALEEC2INSTANCESADISH'. Under the 'DEPLOY' section, 'Test (Ctrl+Shift+I)' is highlighted. In the center, the code editor shows the Python script for scaling EC2 instances. At the bottom, the 'TEST EVENTS' section shows a selected event named 'AUTO-SCALEEC2INSTANC...' with a status of 'Succeeded'. The 'Response' field shows the JSON output of the function execution.

```

# Scale up
if metric_value >= HIGH_THRESHOLD and current_count < MAX_INSTANCES:
    logger.info("High threshold exceeded: %s >= %s", metric_value,
HIGH_THRESHOLD)
    launched = start_new_instance()
    return {"action": "scale_up", "launched": launched}

# Scale down
if metric_value <= LOW_THRESHOLD and current_count > MIN_INSTANCES:
    logger.info("Low threshold met: %s <= %s", metric_value,
LOW_THRESHOLD)
    terminated = terminate_one_instance(managed)
    return {"action": "scale_down", "terminated": terminated}

logger.info("No scaling action required. metric=%s thresholds=(%s,%s)
count=%d", metric_value, LOW_THRESHOLD, HIGH_THRESHOLD, current_count)
return {"action": "no_action", "metric": metric_value}

```