**Question: Can a lambda function detect how many running servers are in my cluster?**

Yes, a Lambda function can detect the number of running servers in your cluster and compare it with the number of subscribers in an SNS topic. If there's a discrepancy, the Lambda can trigger an alert.

Here?s how you can implement this approach:

### Query the Cluster
Use AWS SDK's `DescribeInstances` API to get running EC2 instances or Kubernetes pods for EKS.

#### Example in Scala for EC2
```scala
import software.amazon.awssdk.services.ec2.Ec2Client
import software.amazon.awssdk.services.ec2.model.DescribeInstancesRequest

def getRunningServers: Int = {
    val ec2Client = Ec2Client.create()
    val request = DescribeInstancesRequest.builder()
        .filters(
            Filter.builder().name("tag:AppServer").values("true").build(),
            Filter.builder().name("instance-state-name").values("running").build()
        ).build()

    val instances = ec2Client.describeInstances(request)
    instances.reservations().flatMap(_.instances()).size
```

}

```

### Fetch SNS Subscribers

Use `ListSubscriptionsByTopic` API to fetch the list of subscribers.

#### Example in Scala

```scala
import software.amazon.awssdk.services.sns.SnsClient
import software.amazon.awssdk.services.sns.model.ListSubscriptionsByTopicRequest

def getSnsSubscribers(topicArn: String): Int = {
    val snsClient = SnsClient.create()
    val request = ListSubscriptionsByTopicRequest.builder().topicArn(topicArn).build()
    val subscriptions = snsClient.listSubscriptionsByTopic(request)
    subscriptions.subscriptions().size
}
```

### Compare Counts and Trigger Alerts

Compare server and subscriber counts, and send alerts via SNS or CloudWatch Logs if discrepancies exist.

**Question: These examples are Scala--aren't lambdas written in other languages, like JavaScript?**

Yes, AWS Lambda functions are commonly written in languages like JavaScript (Node.js), Python, or Java, among others.
```

#### Example in JavaScript (Node.js)

```javascript
const { EC2Client, DescribeInstancesCommand } = require("@aws-sdk/client-ec2");
const { SNSClient, ListSubscriptionsByTopicCommand } = require("@aws-sdk/client-sns");

exports.handler = async (event) => {
    const ec2Client = new EC2Client({ region: "us-east-1" });
    const snsClient = new SNSClient({ region: "us-east-1" });

    const getRunningServers = async () => {
        const params = {
            Filters: [
                { Name: "tag:AppServer", Values: ["true"] },
                { Name: "instance-state-name", Values: ["running"] }
            ]
        };
        const data = await ec2Client.send(new DescribeInstancesCommand(params));
        return data.Reservations.flatMap(r => r.Instances).length;
    };

    const getSnsSubscribers = async (topicArn) => {
        const params = { TopicArn: topicArn };
        const data = await snsClient.send(new ListSubscriptionsByTopicCommand(params));
        return data.Subscriptions.length;
    };
```

```
    const runningServers = await getRunningServers();
    const snsSubscribers = await getSnsSubscribers(process.env.TOPIC_ARN);

    if (runningServers !== snsSubscribers) {
                console.error(`Discrepancy detected! Servers: ${runningServers}, Subscribers: ${snsSubscribers}`);
    }
};
```

**Question: Presuming this lambda is running on AWS, rather than a console log, shouldn't it trigger**

Yes, instead of merely logging discrepancies, wire the Lambda function into AWS monitoring services.

### Send Metrics to CloudWatch

Use `PutMetricData` to create custom CloudWatch metrics.

```javascript
const { CloudWatchClient, PutMetricDataCommand } = require("@aws-sdk/client-cloudwatch");

const cloudWatchClient = new CloudWatchClient({ region: "us-east-1" });

const putCloudWatchMetric = async (metricName, value) => {
    const params = {
        Namespace: "ClusterMonitoring",
        MetricData: [{ MetricName: metricName, Value: value, Unit: "Count" }]
    };
```

```
    await cloudWatchClient.send(new PutMetricDataCommand(params));
};

putCloudWatchMetric("DiscrepancyCount", 1);
```

### Trigger Alerts via SNS

Send notifications through SNS.

```javascript
const { SNSClient, PublishCommand } = require("@aws-sdk/client-sns");

const snsClient = new SNSClient({ region: "us-east-1" });
const sendAlert = async (message) => {
    await snsClient.send(new PublishCommand({
        TopicArn: process.env.ALERT_TOPIC_ARN,
        Message: message
    }));
};
sendAlert("Discrepancy detected!");
```

### Use EventBridge for Advanced Monitoring

EventBridge allows integration with other AWS services or workflows for more sophisticated alerting.