Challenge 4 - Welcome to management

Background

As mentioned in Challenge 3, a cluster allows you to minimize the friction that often comes with deployments, scaling, and upgrades. However, initiating these processes manually via `kubectl` commands or Service Fabric PowerShell scripts can begin to feel just as tedious. Your players also have no way to lookup your servers.

Challenge

Your team's goal in this challenge is to create a REST API to: list, add and delete instances from your cluster **and** a web application which provides an intuitive visual interface on top of your API for administrators to manage your cluster and players to lookup available servers.

You should also ensure that your telemetry and monitoring solution is kept up to date, and is integrated into web application in a useful way.

The REST API for listing servers should return a JSON array of tenants with the following properties. You may include additional properties in the response as desired, but they will be ignored by the automated verification:

```
[
  {
    "name": "<some arbitrary name>",
    "endpoints": {
      "minecraft": "<publicly available IP:port>",
      "rcon": "<publicly available IP:port>"
    }
  }
]
```

An example response might look like:

```
[
  {
    "name": "tenant1",
    "endpoints": {
      "minecraft": "128.124.90.15:25565",
      "rcon": "128.124.90.15:25575"
    }
  },
  {
    "name": "tenant2",
    "endpoints": {
      "minecraft": "128.194.90.16:25565",
      "rcon": "128.194.90.16:25575"
    }
  },
  {
    "name": "tenant3",
    "endpoints": {
      "minecraft": "128.194.90.16:25566",
      "rcon": "128.194.90.16:25576"
    }
  }
]
```

Success Criteria

- Your REST API should be able to add and remove instances from your cluster.
- Your REST API should return a list of servers that meets the above specification.
- Your web application should allow players to view a list of available servers, with relevant information to allow them to connect directly.
- Your web application should have an administrative interface that the cluster and instances can be managed from.
- Submit an endpoint for your REST API server-list to the OpenHack portal. The portal will verify that the response meets our specification and that connections can be made to the returned endpoints, *Ensure you have scaled your cluster so that your API returns more than 1 instance.*
- Demonstrate your web application to a coach, and be sure to point out the management functionality, and telemetry and monitoring options you have included

telemetry and monitoring options you have included.

References

- Hint: there are no points for style, but it helps!
- docs.microsoft.com Is a great place to start considering options for your solution here.
- Azure Functions offer an alternative approach for some functionality that may be useful in this or *future* challenges.