Azure Resource Groups with Python

Introduction

I'll guide you through achieving the same resource group tasks using Python with the Azure SDK. Based on your Azure Guide document, it seems you already have the necessary setup to get started.

Step-by-Step Guide for Azure Resource Groups with Python

Prerequisites

First, make sure you have the required packages installed:

```
pip install azure-identity azure-mgmt-resource
```

1. List Resource Groups in a Subscription

Let's create a Python script to list all resource groups:

```
from azure.identity import AzureCliCredential
from azure.mgmt.resource import ResourceManagementClient

# Authenticate using Azure CLI credentials
credential = AzureCliCredential()

# Get subscription ID
from azure.mgmt.resource import SubscriptionClient
subscription_client = SubscriptionClient(credential)
subscription = next(subscription_client.subscriptions.list())
subscription_id = subscription.subscription_id

print(f"Using subscription: {subscription.display_name}
({subscription_id})")

# Create a Resource Management client
resource_client = ResourceManagementClient(credential, subscription_id)

# List all resource groups
print("\nResource Groups:")
for group in resource_client.resource_groups.list():
    print(f"- {group.name} (Location: {group.location})")
```

2. Create a New Resource Group

Add the following code to create a new resource group:

```
def create_resource_group(resource_client, name, location="eastus"):
    """Create a new resource group"""
    print(f"\nCreating Resource Group '{name}' in {location}...")

    # Create the resource group
    result = resource_client.resource_groups.create_or_update(
        name,
        {"location": location}
    )

    print(f"Resource Group '{result.name}' created successfully in
{result.location}")
    return result

# Example usage
new_group = create_resource_group(resource_client, "MyNewResourceGroup",
"westus")
```

3. List Resources in a Resource Group

Now, let's add code to list all resources in a specified resource group:

```
def list_resources_in_group(resource_client, group_name):
    """List all resources in a specific resource group"""
    print(f"\nResources in '{group_name}':")

    # Get all resources in the group
    resources =
resource_client.resources.list_by_resource_group(group_name)

    # Display each resource
    found = False
    for resource in resources:
        found = True
        print(f"- Name: {resource.name}")
        print(f"  Type: {resource.type}")
        print(f"  Location: {resource.location}")
        print(f"  ID: {resource.id}")
        print()

    if not found:
        print("No resources found in this group.")

# Example usage
list_resources_in_group(resource_client, "MyNewResourceGroup")
```

4. Complete Script

Here's a complete, reusable script that combines all these functionalities:

```
from azure.identity import AzureCliCredential
from azure.mgmt.resource import ResourceManagementClient,
SubscriptionClient

def main():
    # Authenticate using Azure CLI credentials
    credential = AzureCliCredential()

    # Get subscription ID
    subscription_client = SubscriptionClient(credential)
    subscription = next(subscription_client.subscriptions.list())
    subscription_id = subscription.subscription_id

    print(f"Using subscription: {subscription.display_name}
({subscription_id})")

    # Create Resource Management client
    resource_client = ResourceManagementClient(credential,
subscription_id)

    # List all resource groups
    list_resource_groups(resource_client)

    # Create a new resource group
    group_name = "PythonResourceGroup"
    create_resource_group(resource_client, group_name, "eastus")

    # List resources in the new group
    list_resources_in_group(resource_client, group_name)

def list_resource_groups(resource_client):
    """List all resource groups in the subscription"""
    print("\nResource Groups:")
    for group in resource_client.resource_groups.list():
        print(f"- {group.name} (Location: {group.location})")

def create_resource_group(resource_client, name, location="eastus"):
    """Create a new resource group"""
    print(f"\nCreating Resource Group '{name}' in {location}...")

    # Create the resource group
    result = resource_client.resource_groups.create_or_update(
        name,
        {"location": location}
    )

    print(f"Resource Group '{result.name}' created successfully in
{result.location}")
    return result

def list_resources_in_group(resource_client, group_name):
    """List all resources in a specific resource group"""
```

```
    print(f"\nResources in '{group_name}':")

    # Get all resources in the group
    resources =
resource_client.resources.list_by_resource_group(group_name)

    # Display each resource
    found = False
    for resource in resources:
        found = True
        print(f"- Name: {resource.name}")
        print(f"  Type: {resource.type}")
        print(f"  Location: {resource.location}")
        print()

    if not found:
        print("No resources found in this group.")

if __name__ == "__main__":
    main()
```

5. Additional Functionality - Working with Specific Subscriptions

If you want to choose a specific subscription, you can modify the script:

```
def select_subscription(credential):
    """Allow user to select a subscription"""
    subscription_client = SubscriptionClient(credential)

    # List all subscriptions
    print("\nAvailable Subscriptions:")
    subscriptions = list(subscription_client.subscriptions.list())

    for i, sub in enumerate(subscriptions):
        print(f"{i+1}. {sub.display_name} ({sub.subscription_id})")

    # Let user select a subscription
    choice = int(input("\nSelect a subscription (number): ")) - 1
    selected_subscription = subscriptions[choice]

    print(f"Selected: {selected_subscription.display_name}")
    return selected_subscription.subscription_id
```

Then use it in your main function like this:

```
# Get user-selected subscription
subscription_id = select_subscription(credential)
```

```
# Create Resource Management client with selected subscription
resource_client = ResourceManagementClient(credential, subscription_id)
```

This script gives you a solid foundation for working with Azure Resource Groups in Python. You can save it as `azure_resource_manager.py` and run it with `python azure_resource_manager.py`.