Security Appliance

Realisation document

1. Introduction

This document showcases all the realisation I made towards increasing the security of the Azure infrastructure of the OpenRemotes pipeline. All steps were previously discussed in the security report. It is very important to ensure maximum security in every environment to lessen all security risks. In the next paragraphs, I will explain every action I have made in every scope of the security report.

2. Implementation

2.1. Identity and Access Management

For the IAM security requirements, I have set up an automatic administrator account creation that enforces MFA in 14 days. As I described in the security report, it is not advised to work and access any cloud platform from the root account. This administrator account will allow access to the Azure environment more securely. For the pipeline-specific role, I have decided to hold off because we have to ensure everything is working correctly in the pipeline as I have encountered a few issues.

iam.tf:

```
data "azuread_domains" "default_domain" {
  only_default = true
}

You, 5 days ago | 1 author (You)
resource "random_password" "random_admin_password" {
  count   = var.enable_admin_account ? 1 : 0
  length  = 16
  special = true
}

You, 5 days ago | 1 author (You)
resource "azuread_user" "admin_user" {
  count = var.enable_admin_account ? 1 : 0

  user_principal_name   = "admin@${data.azuread_domains.default_domain.domains[0].domain_name}"
  display_name          = "OpenRemote Admin"
  mail_nickname         = "admin"
  password              = random_password.random_admin_password[count.index].result
  force_password_change = true
}
```

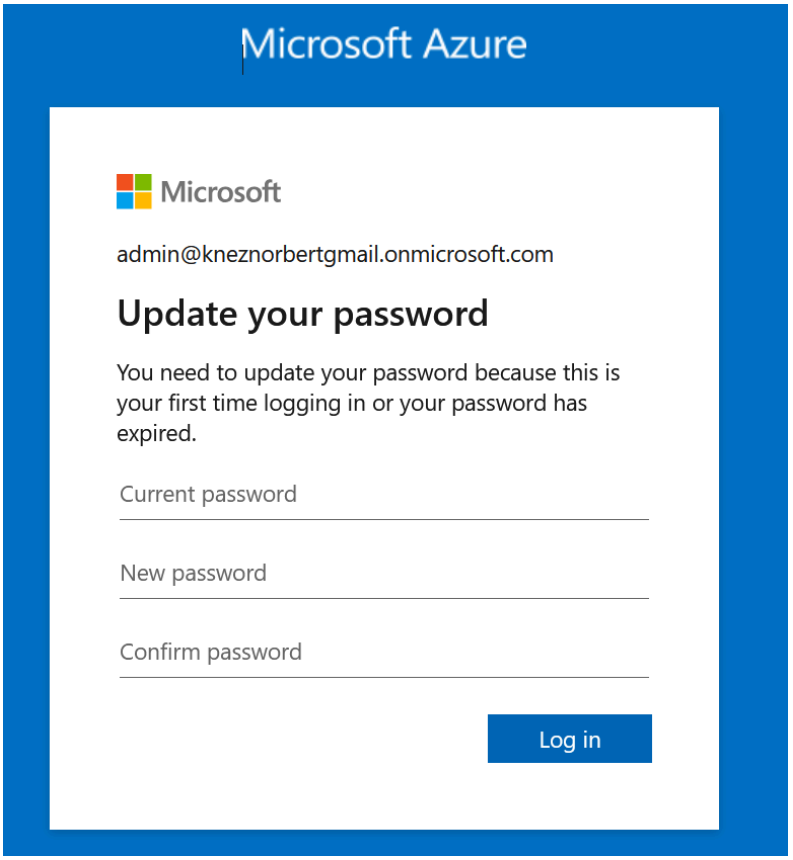
First, terraform reads the default domain of the Azure account for the email of admin account purposes. Then it generates a random password for that account. Lastly, I am creating an admin account that consists of the email of admin + data terraform read earlier, the display name of “OpenRemote Admin”, the nickname of “admin”, and the password that we generated earlier and force change it on the next login.

```hcl
resource "azurerm_role_assignment" "admin_user_assignment" {
  count = var.enable_admin_account ? 1 : 0

  scope                = azurerm_resource_group.openremote-rg.id
  role_definition_name = "Owner"
  principal_id         = azuread_user.admin_user[count.index].object_id
}

You, 5 days ago | 1 author (You)
output "admin_credentials" {
  value = var.enable_admin_account ? {
    username = azuread_user.admin_user[0].user_principal_name
    password = random_password.random_admin_password[0].result
  } : null
  sensitive = true
}
```

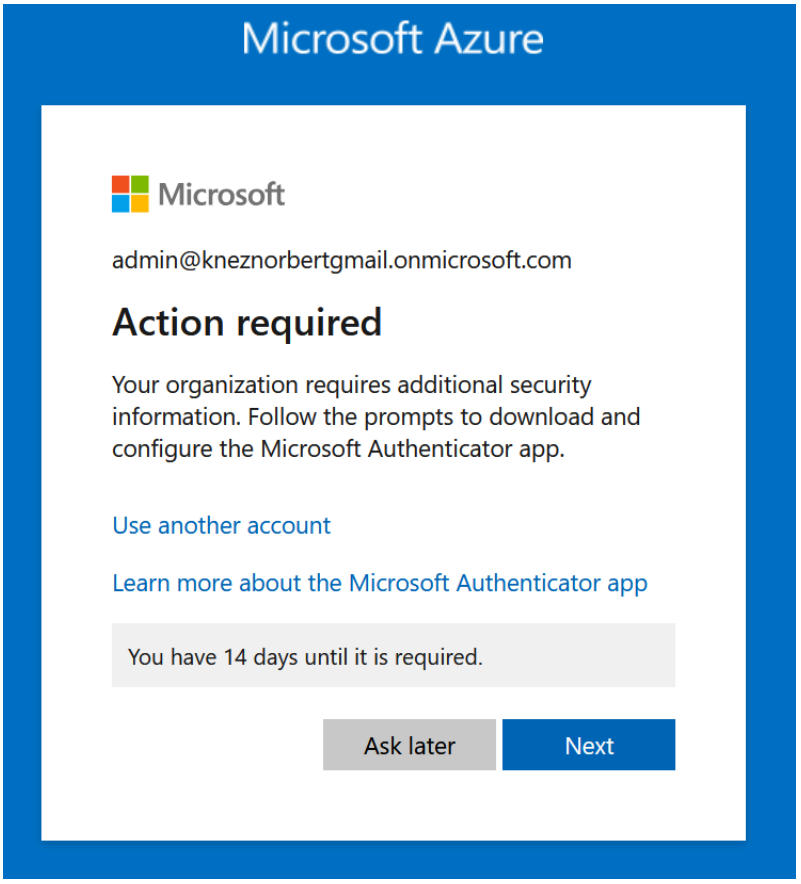
On this screenshot, I am assigning Owner permissions to the administrator account and then outputting in the console all necessary information about the account like username and password.

```
nknez@NKLEGION:~/GitProjects/openremote-azure-pipeline/terraform-azure$ terraform output admin_credentials
{
  "password" = [redacted]
  "username" = "admin@kneznorbertgmail.onmicrosoft.com"
}
```

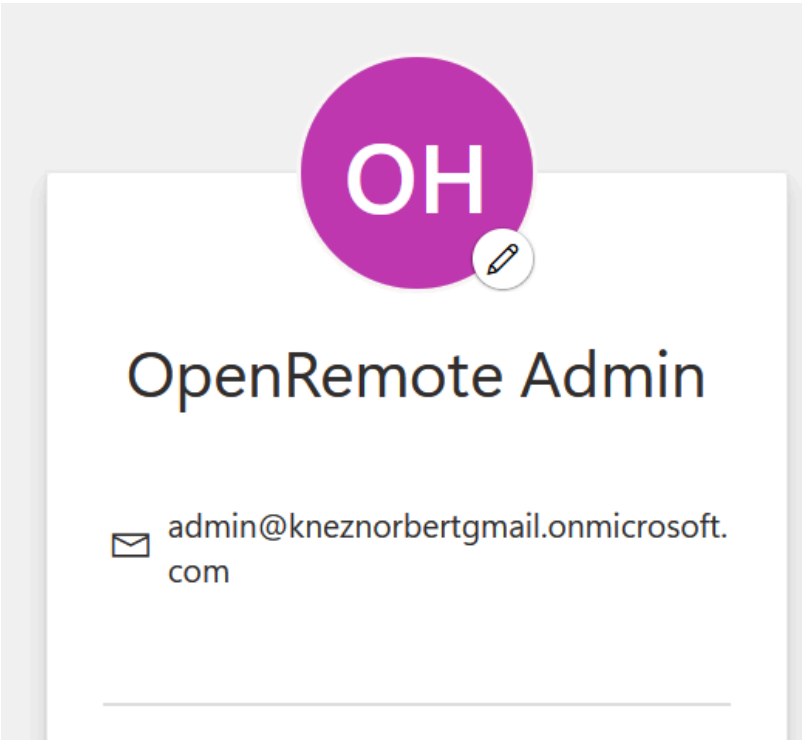
Output that consists of the details of the admin account after deploying the infrastructure.



After logging in, you are asked to set up a new password.



After the password change, you are forced to set up a Multi-Factor Authentication within 14 days.



Account dashboard details.

2.2. Network Security

To improve the network security, I have decided to move the Virtual Machine to the private subnet, implement NAT Gateway to give it access to the Internet and set up a load balancer for accessing the IoT platform from the Internet. Also, restricting SSH port to only be accessible from the Azure Virtual Network and setting up Azure Bastion as a way to connect to the Virtual Machine. All that architecture increases the security of the VM as the ports of it are not exposed to the public internet, only to the load balancer. I decided to not implement Azure DDoS protection as this service is way too overpriced for this simple architecture.

loadbalancer.tf:

```
resource "azurerm_public_ip" "openremote-lb-ip" {
  count               = var.enable_private_vm_setup ? 1 : 0
  name                = "openremote-lb-ip"
  resource_group_name = azurerm_resource_group.openremote-rg.name
  location            = azurerm_resource_group.openremote-rg.location
  allocation_method   = "Static"
  sku                 = "Standard"
}

You, 4 days ago | 1 author (You)
resource "azurerm_lb" "load_balancer" {
  count               = var.enable_private_vm_setup ? 1 : 0
  name                = "openremote-lb"
  location            = azurerm_resource_group.openremote-rg.location
  resource_group_name = azurerm_resource_group.openremote-rg.name
  sku                 = "Standard"

  You, 4 days ago | 1 author (You)
  frontend_ip_configuration {
    name                 = "frontend"
    public_ip_address_id = azurerm_public_ip.openremote-lb-ip[count.index].id
  }
}

You, 4 days ago | 1 author (You)
resource "azurerm_lb_backend_address_pool" "openremote-lb-pool" {
  count           = var.enable_private_vm_setup ? 1 : 0
  name            = "openremote-lb-pool"
  loadbalancer_id = azurerm_lb.load_balancer[count.index].id
}
```

Here I have created an IP address for the load balancer as it is necessary for accessing the VM. Then created the load balancer within the resource group and created the frontend configuration with the IP I had created before. Then I have to create a backend address pool in which I can add VM later on.

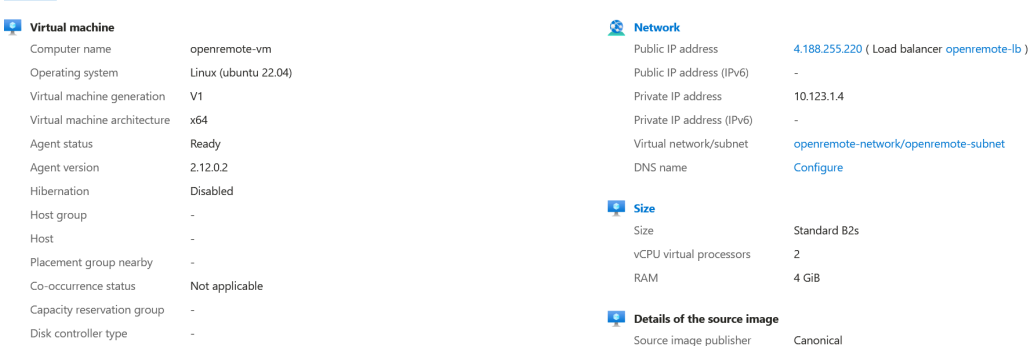
```
resource "azurerm_lb_rule" "https_rule" {
  count                          = var.enable_private_vm_setup ? 1 : 0
  name                           = "https-rule"
  loadbalancer_id                = azurerm_lb.load_balancer[count.index].id
  frontend_ip_configuration_name = "frontend"
  protocol                       = "Tcp"
  frontend_port                  = 443
  backend_port                   = 443
  backend_address_pool_ids       = [azurerm_lb_backend_address_pool.openremote-lb-pool[count.index].id]
  probe_id                       = azurerm_lb_probe.https_probe[count.index].id
  disable_outbound_snat          = true
}

You, 4 days ago | 1 author (You)
resource "azurerm_lb_rule" "http_rule" {
  count                          = var.enable_private_vm_setup ? 1 : 0
  name                           = "http-rule"
  loadbalancer_id                = azurerm_lb.load_balancer[count.index].id
  frontend_ip_configuration_name = "frontend"
  protocol                       = "Tcp"
  frontend_port                  = 80
  backend_port                   = 80
  backend_address_pool_ids       = [azurerm_lb_backend_address_pool.openremote-lb-pool[count.index].id]
  probe_id                       = azurerm_lb_probe.http_probe[count.index].id
  disable_outbound_snat          = true
}
```

```
resource "azurerm_lb_rule" "mqtt_rule" {
  count                          = var.enable_private_vm_setup ? 1 : 0
  name                           = "mqtt-rule"
  loadbalancer_id                = azurerm_lb.load_balancer[count.index].id
  frontend_ip_configuration_name = "frontend"
  protocol                       = "Tcp"
  frontend_port                  = 8883
  backend_port                   = 8883
  backend_address_pool_ids       = [azurerm_lb_backend_address_pool.openremote-lb-pool[count.index].id]
  disable_outbound_snat          = true
}

You, 4 days ago | 1 author (You)
resource "azurerm_lb_rule" "smtp_rule" {
  count                          = var.enable_private_vm_setup ? 1 : 0
  name                           = "smtp-rule"
  loadbalancer_id                = azurerm_lb.load_balancer[count.index].id
  frontend_ip_configuration_name = "frontend"
  protocol                       = "Tcp"
  frontend_port                  = 25
  backend_port                   = 25
  backend_address_pool_ids       = [azurerm_lb_backend_address_pool.openremote-lb-pool[count.index].id]
  disable_outbound_snat          = true
}
```

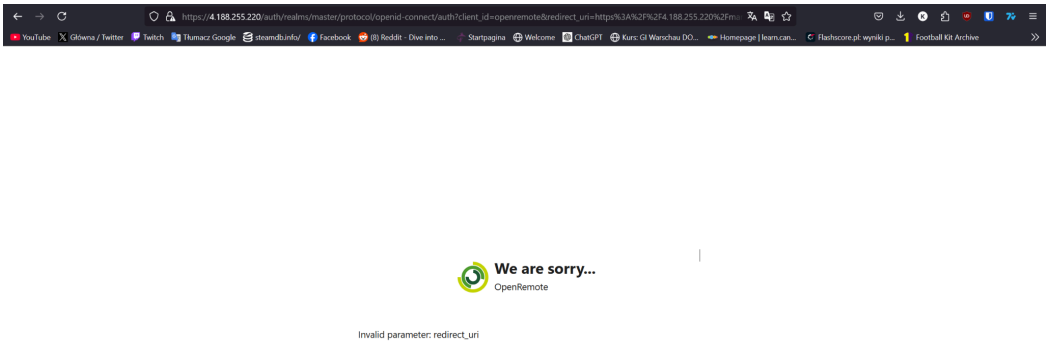
Here I am exposing HTTP, HTTPS, MQTT and SMTP ports on the load balancer as the IoT platform requires them to work properly.

```hcl
resource "azurerm_lb_probe" "http_probe" {
  count           = var.enable_private_vm_setup ? 1 : 0
  name            = "http-probe"
  loadbalancer_id = azurerm_lb.load_balancer[count.index].id
  protocol        = "Tcp"
  port            = 80
}

You, 4 days ago | 1 author (You)
resource "azurerm_lb_probe" "https_probe" {
  count           = var.enable_private_vm_setup ? 1 : 0
  name            = "https-probe"
  loadbalancer_id = azurerm_lb.load_balancer[count.index].id
  protocol        = "Tcp"
  port            = 443
}
```

Here I am creating two health checks for the VM on ports 80 and 443. Thanks to this, when a VM has issues or goes offline, the load balancer won't redirect connections to it.



Virtual Machine dashboard. You can its public IP address is load balancers.



After typing the IP of the load balancer you can access IoT platform (currently having a bug but that will be fixed)

nat.tf:

```
resource "azurerm_public_ip" "nat_gw_ip" {
  count               = var.enable_private_vm_setup ? 1 : 0
  name                = "nat-gw-ip"
  resource_group_name = azurerm_resource_group.openremote-rg.name
  location            = azurerm_resource_group.openremote-rg.location
  sku                 = "Standard"
  allocation_method   = "Static"
}

You, 4 days ago | 1 author (You)
resource "azurerm_nat_gateway" "openremote_nat_gw" {
  count               = var.enable_private_vm_setup ? 1 : 0
  name                = "openremote-nat-gw"
  location            = azurerm_resource_group.openremote-rg.location
  resource_group_name = azurerm_resource_group.openremote-rg.name
  sku_name            = "Standard"
}

You, 4 days ago | 1 author (You)
resource "azurerm_nat_gateway_public_ip_association" "name" {
  count                = var.enable_private_vm_setup ? 1 : 0
  nat_gateway_id       = azurerm_nat_gateway.openremote_nat_gw[count.index].id
  public_ip_address_id = azurerm_public_ip.nat_gw_ip[count.index].id
}

You, 4 days ago | 1 author (You)
resource "azurerm_subnet_nat_gateway_association" "openremote_subnet_nat_gw_assoc" {
  count          = var.enable_private_vm_setup ? 1 : 0
  subnet_id      = azurerm_subnet.openremote-subnet.id
  nat_gateway_id = azurerm_nat_gateway.openremote_nat_gw[count.index].id
}
```

Here I have created an IP address for the NAT Gateway, creating NAT Gateway in the same resource groups as other resources, associating the IP address with it, and putting the NAT Gateway in the same subnet as the Virtual Machine so it has the necessary Internet connection.

main.tf:

```
resource "azurerm_network_security_rule" "openremote-dev-rule" {
  depends_on = [
    azurerm_network_security_group.openremote-sg
  ]
  name                        = "openremote-dev-rule"
  priority                    = 100
  direction                   = "Inbound"
  access                      = "Allow"
  protocol                    = "*"
  source_port_range           = "*"
  destination_port_range      = "22"
  source_address_prefix       = var.enable_private_vm_setup ? "VirtualNetwork" : var.ssh_source_ip
  destination_address_prefix  = "*"
  resource_group_name         = azurerm_resource_group.openremote-rg.name
  network_security_group_name = azurerm_network_security_group.openremote-sg.name
}
```

I have adjusted some of my teammates code to make it work with my new architecture. Here I have implemented dependant SSH source dependent on the variable. So either it is a public IP of the client or the whole Virtual Network in Azure.

```
resource "azurerm_network_security_rule" "openremote-mqtt" {
  name                        = "openremote-mqtt"
  priority                    = 103
  direction                   = "Inbound"
  access                      = "Allow"
  protocol                    = "Tcp"
  source_port_range           = "*"
  destination_port_range      = "8883"
  source_address_prefix       = "*"
  destination_address_prefix  = "*"
  network_security_group_name = azurerm_network_security_group.openremote-sg.name
  resource_group_name         = azurerm_resource_group.openremote-rg.name
}

You, 4 days ago | 1 author (You)
resource "azurerm_network_security_rule" "openremote-smtp" {
  name                        = "openremote-smtp"
  priority                    = 104
  direction                   = "Inbound"
  access                      = "Allow"
  protocol                    = "Tcp"
  source_port_range           = "*"
  destination_port_range      = "25"
  source_address_prefix       = "*"
  destination_address_prefix  = "*"
  network_security_group_name = azurerm_network_security_group.openremote-sg.name
  resource_group_name         = azurerm_resource_group.openremote-rg.name
}
```

I have exposed MQTT and SMTP ports for the Virtual Machine as they are necessary for some of the IoT platform functionality.

```
resource "azurerm_public_ip" "openremote-ip" {
  count               = var.enable_private_vm_setup ? 0 : 1
  name                = "openremote-ip"
  resource_group_name = azurerm_resource_group.openremote-rg.name
  location            = azurerm_resource_group.openremote-rg.location
  allocation_method   = "Dynamic"
  sku                 = "Basic"
}

You, 4 days ago | 2 authors (rubyfeller and one other)
resource "azurerm_network_interface" "openremote-nic" {
  name                = "openremote-nic"
  location            = azurerm_resource_group.openremote-rg.location
  resource_group_name = azurerm_resource_group.openremote-rg.name

  You, 4 days ago | 2 authors (rubyfeller and one other)
  ip_configuration {
    name                          = "internal"
    subnet_id                     = azurerm_subnet.openremote-subnet.id
    private_ip_address_allocation = "Dynamic"
    public_ip_address_id          = var.enable_private_vm_setup ? null : azurerm_public_ip.openremote-ip[0].id
  }
}
```

Here I have added some dependent variables to the already existing code (I will explain them later)

```
resource "azurerm_network_interface_backend_address_pool_association" "openremote-nic-backend-pool" {
  count                   = var.enable_private_vm_setup ? 1 : 0
  network_interface_id    = azurerm_network_interface.openremote-nic.id
  ip_configuration_name   = azurerm_network_interface.openremote-nic.ip_configuration[0].name
  backend_address_pool_id = azurerm_lb_backend_address_pool.openremote-lb-pool[count.index].id
}
```

This is associating the Virtual Machine with the backend address pool of the load balancer.

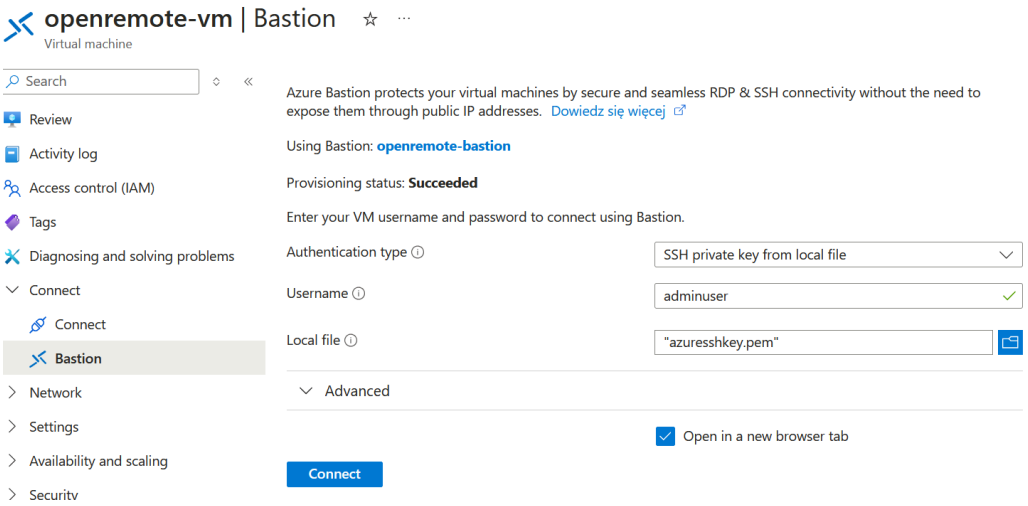
```
output "instance_details" {
  value = var.enable_private_vm_setup ? {
    name       = azurerm_linux_virtual_machine.openremote-vm.name
    private_ip = azurerm_network_interface.openremote-nic.private_ip_address
    } : {
    name       = azurerm_linux_virtual_machine.openremote-vm.name
    private_ip = azurerm_network_interface.openremote-nic.private_ip_address
  }
}
```

I did a little enchantment to the output of instance details.

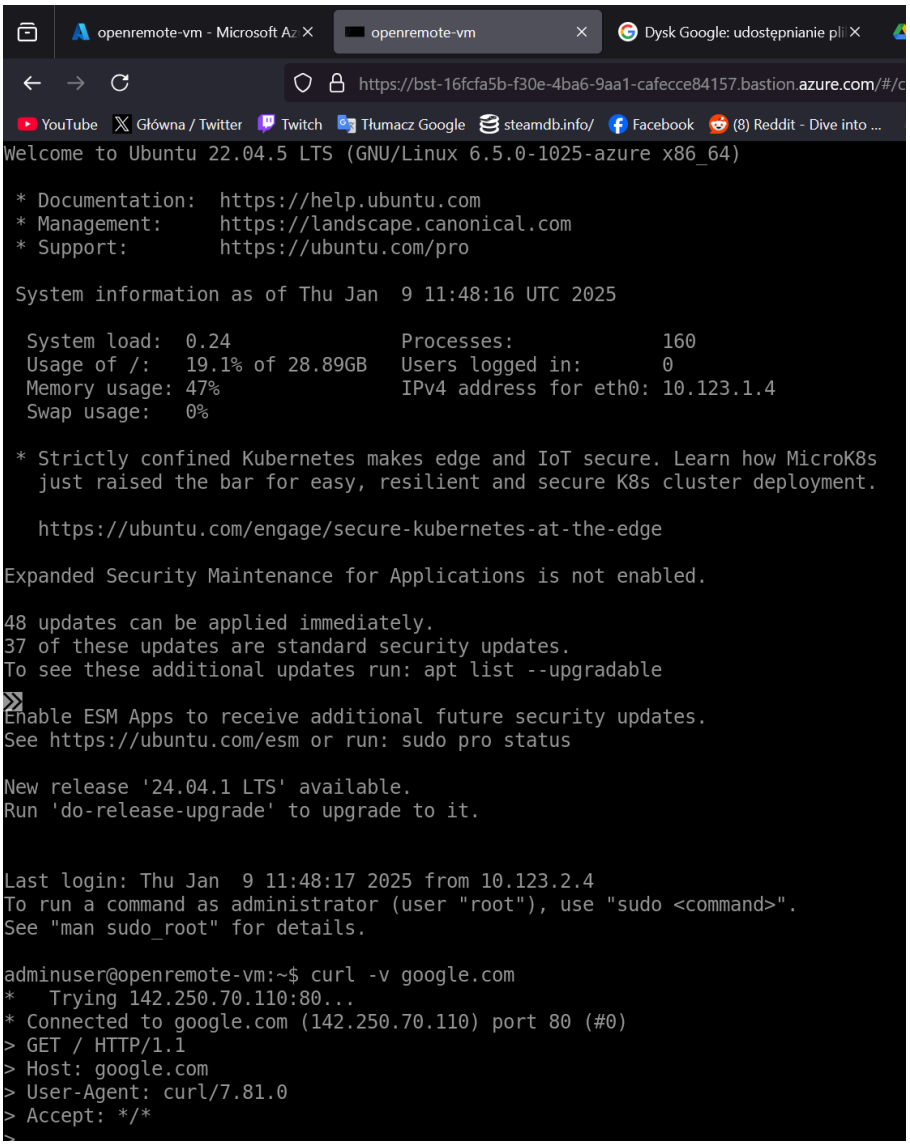
bastion.tf:

```
resource "azurerm_subnet" "bastion_subnet" {
  count                = var.enable_private_vm_setup ? 1 : 0
  name                 = "AzureBastionSubnet"
  resource_group_name  = azurerm_resource_group.openremote-rg.name
  virtual_network_name = azurerm_virtual_network.openremote-vn.name
  address_prefixes     = ["10.123.2.0/24"]
}


You, 4 days ago | 1 author (You)
resource "azurerm_public_ip" "bastion_public_ip" {
  count               = var.enable_private_vm_setup ? 1 : 0
  name                = "openremote-bastion-ip"
  resource_group_name = azurerm_resource_group.openremote-rg.name
  location            = azurerm_resource_group.openremote-rg.location
  allocation_method   = "Static"
  sku                 = "Standard"
}

You, 4 days ago | 1 author (You)
resource "azurerm_bastion_host" "bastion" {
  count               = var.enable_private_vm_setup ? 1 : 0
  name                = "openremote-bastion"
  location            = azurerm_resource_group.openremote-rg.location
  resource_group_name = azurerm_resource_group.openremote-rg.name       You, 5 days

  You, 4 days ago | 1 author (You)
  ip_configuration {
    name                 = "ipconfig"
    subnet_id            = azurerm_subnet.bastion_subnet[count.index].id
    public_ip_address_id = azurerm_public_ip.bastion_public_ip[count.index].id
  }
}
```

Here are all Azure Bastion settings. First I am creating a separate subnet for it and also an IP address for it. Then I am creating the Bastion with all necessary configurations like public IP, subnet and resource group.



Bastion dashboard on Azure website. You can connect to the Virtual Machine through a specific SSH key earlier defined in the setup.



Connected successfully to the Virtual Machine. You can see here that it has Internet access thanks to the NAT Gateway.

2.3. Data Protection

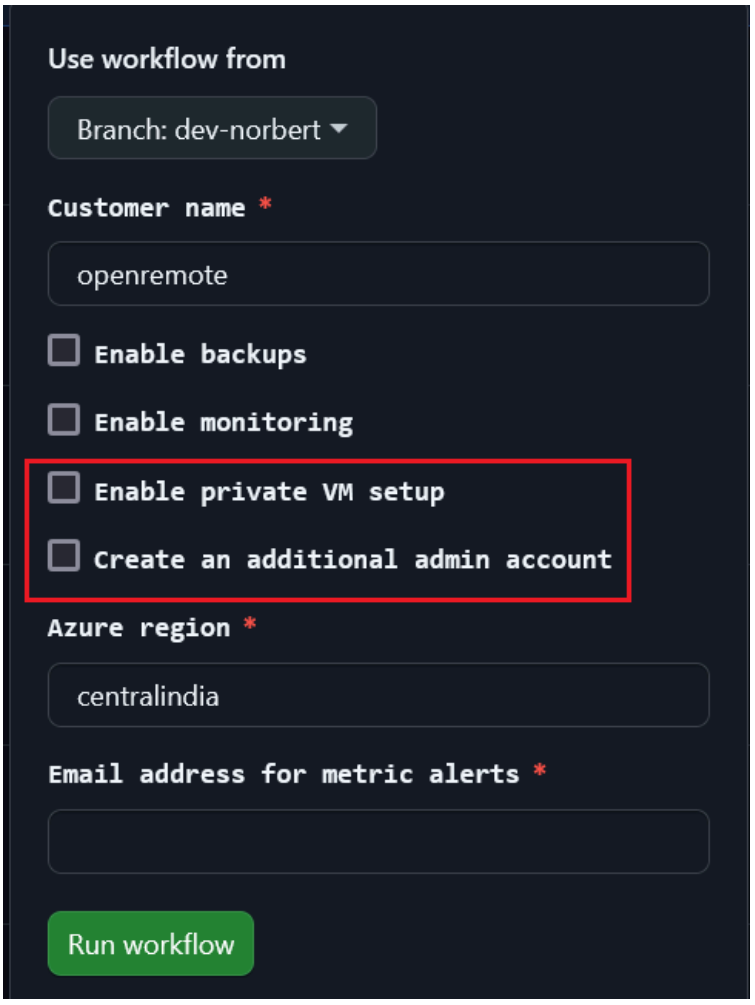
With encrypting the Terraform file state, I have decided to hold off as I was focusing on the network and IAM security. This change is less important than the rest and the time was a big issue so this change could be implemented in the future

2.4. Monitoring and Backups

. The monitoring and backup changes were implemented by my teammate. He improved the monitoring by implementing much more necessary metrics.

2.5. Pipeline changes

I have implemented a few minor changes to pipeline yaml files. Mostly they are about new checkboxes and dependable variables to choose which architecture you want to deploy. I can imagine not everyone needs an additional administrator account or extra-secure private VM architecture as this could add up some costs and admin overhead so this adds nice a way to choose.

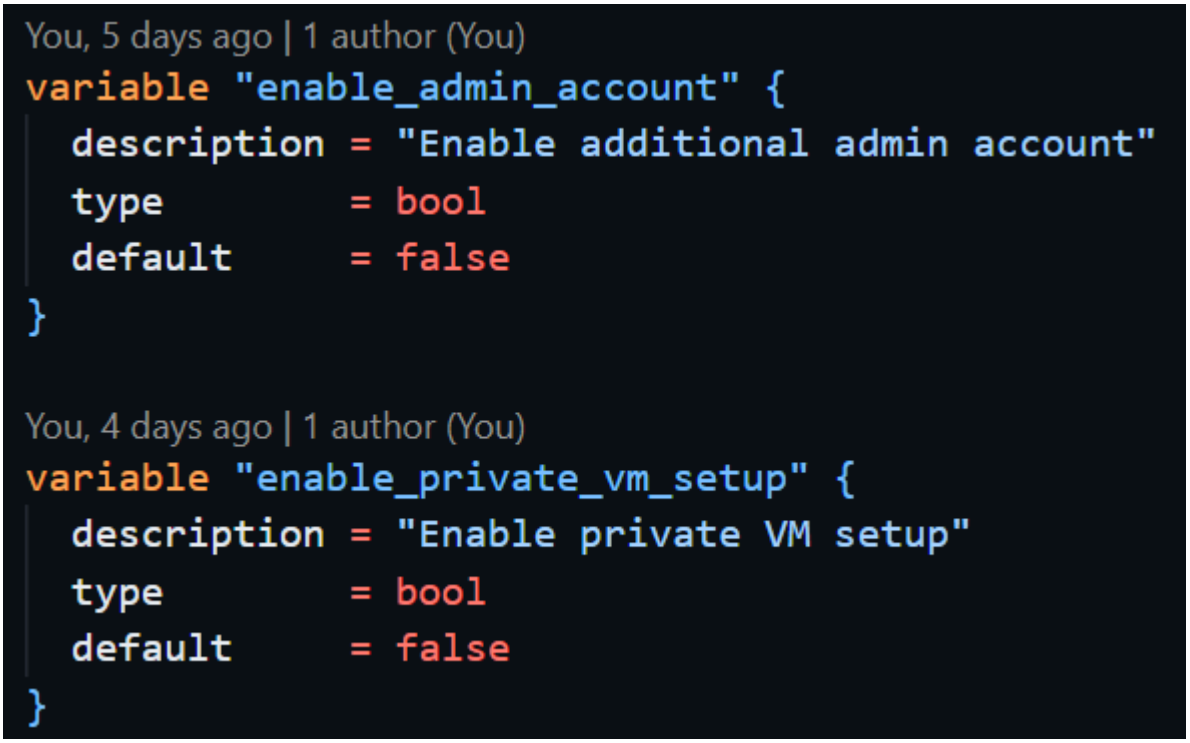


To the pipeline workflow dispatch menu, I have added two new variables: **Enable private VM setup** and **Create an additional admin account**.

Enable private VM setup - this enables the more secure architecture consisting of a load balancer, virtual machine in private subnet and nat gateway instead of just a virtual machine with a public IP address.

Create an additional admin account - as the name suggests, it is creating an alternative account to avoid working in the root account for increased security measures.

Inside the Terraform I have implemented two variables:



They are necessary to make the choice in the pipeline work and they enable to creation of specific resources.

Example usage of the variables:

```
resource "azurerm_subnet" "bastion_subnet" {
  count                = var.enable_private_vm_setup ? 1 : 0
  name                 = "AzureBastionSubnet"
  resource_group_name  = azurerm_resource_group.openremote-rg.name
  virtual_network_name = azurerm_virtual_network.openremote-vn.name
  address_prefixes     = ["10.123.2.0/24"]
}
```

```
resource "random_password" "random_admin_password" {
  count   = var.enable_admin_account ? 1 : 0
  length  = 16
  special = true
}
```

Here if the passed variable is true (as they are boolean) it will create the resource, otherwise, it will not and avoid creating unnecessary resources.

3. Conclusion

This document showcases all the work I did to improve the security of the Azure Pipeline project. I focused on making key upgrades like better identity management, improving network security, and adding more flexibility to the pipeline setup. Some security measures I did not implement because of deadlines, but they can be added later if needed.

Overall, the changes I made should give the project a solid level of security while keeping it practical and not too expensive. As the semester wraps up, I think this work strikes a good balance between strengthening security and being easy to manage in the long run.