**AWS Session Manager Implementation Plan Across Environments**

**Objective**

Migrate from SSH/PEM-based EC2 access to AWS Systems Manager Session Manager across all AWS accounts, starting with **mwt-internal** and expanding other accounts. This implementation applies to both **Linux** and **Windows** instances, and leverages **IAM Identity Center** (AWS SSO) for access control.

**Benefits**

* No SSH keys or bastion hosts
* IAM-based access control via permission sets
* Logging and auditing through CloudWatch or S3
* No need for public IPs or open inbound ports
* Works in private subnets via VPC endpoints

**General Architecture Requirements**

* EC2 instances with latest **SSM Agent** (Linux & Windows)
* IAM role attached to EC2: **AmazonSSMManagedInstanceCore**
* 3 VPC Interface Endpoints:
  + **ssm**
  + **ssmmessages**
  + **ec2messages**
* Security Group for VPC Endpoints:
  + Inbound: HTTPS (443) from EC2 subnet or SG
  + Outbound: All traffic
* IAM Identity Center permission set granting Session Manager permissions
* (Optional) CloudWatch or S3 logging for session tracking

**Phase 1: Implementation in mwt-internal**

1. **Enable SSM in Linux EC2**

* Attach IAM Role with **AmazonSSMManagedInstanceCore**
* Confirm or install SSM Agent:
  + sudo systemctl status amazon-ssm-agent
  + sudo yum install -y amazon-ssm-agent
  + sudo systemctl enable --now amazon-ssm-agent

1. **Enable SSM in Windows EC2**

* Attach IAM Role with **AmazonSSMManagedInstanceCore**
* Confirm or install SSM Agent:
  + Get-Service AmazonSSMAgent
  + Start-Service AmazonSSMAgent
  + Set-Service AmazonSSMAgent -StartupType Automatic

1. **Create VPC Endpoints (one-time per VPC/region)**

* Interface **endpoints**:
  + com.amazonaws.<region>.ssm
  + com.amazonaws.<region>.ssmmessages
  + com.amazonaws.<region>.ec2messages
* **Subnets**: Use EC2 subnets
* Security Group:
  + Inbound HTTPS from EC2 subnet CIDR

1. **Configure IAM Identity Center**

* Update Permission Set with:

{

"Effect": "Allow",

"Action": [

"ssm:StartSession",

"ssm:DescribeSessions",

"ssm:TerminateSession",

"ssm:GetConnectionStatus",

"ssm:DescribeInstanceInformation",

"ec2:DescribeInstances"

],

"Resource": "\*"

}

1. **Verify connectivity (Linux)**

* Use Session Manager Console (or CLI):
  + aws ssm start-session --target <instance-id>

1. **RDP Support for Windows via Port Forwarding**

* aws ssm start-session \

--target <instance-id> \

--document-name AWS-StartPortForwardingSession \

--parameters "portNumber=3389,localPortNumber=13389"

* Create .rdp file with:

full address:s:127.0.0.1:13389

username:s:Administrator

**Automatable with Terraform (Resource creation on AWS)**

* VPC Interface Endpoints
* Security Group for Endpoints
* IAM Role for EC2
* EC2 Instance Profile

**Automatable with Ansible (VM Configuration)**

Prepare an Ansible playbook that:

* Installs the SSM Agent if it’s missing
* Restarts it if it’s already installed

NOTE: this script will support both linux and windows / Or we will create 2 scripts it depends